1991

A solid-state electronic two-tap weight linear adaptive neuron

Chun-Yu Malcolm Chen
Lehigh University

A SOLID-STATE ELECTRONIC TWO-TAP WEIGHT LINEAR ADAPTIVE NEURON

by

Chun-Yu Malcolm Chen

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

September 24, 1990

Certificate of Approval

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

September 23, 1990
(date)

Warren H. White
Professor in Charge

Lawrence J. Vanasse
Chairman of Department

Acknowledgments

The author wishes to express his immense gratitude and respect to Dr. Marvin H. White, for his guidance and support. Without his indispensable advice and encouragement, this work would not be possible. The author would like to extend his appreciation to Richard Siergiej for his ever ready help and patience to my endless questions. A special word of thanks to Peggy French for providing the SONOS transistor samples. It is an opportunity to acknowledge the author's fellow colleagues, Sukyoon Yoon, Yin Hu, Zhigang Ma, and Paul Orphanos for creating and sharing a harmonic working environment and for their valuable comments.

The author treasures the precious discussions of Dr. Frank Libsch, Dr. Anirban Roy, Dr. Thomas Krutsick, and Dr. Umesh Sharma during his first year of graduate studies. The author is grateful to Dr. Richard Booth for providing software package which is used in part of the thesis. The contributions of a summer student, Harikaran Sathianathan, have been most helpful.

The author is indebted to the Office of Naval Research (ONR) and the Defense Advanced Research Project Agency (DARPA) Artificial Intelligence Neural Networks Technology Program for funding the work at the Sherman Fairchild Center. Fellowship support from the National Science Foundation (NSF) Engineering Research Center of Advanced Technology for Large Structural Systems (ATLSS) at Lehigh University is greatly appreciated.

This work is dedicated to the parents of the author for their everlasting love and support and their spiritual influence toward excellence.

Table of Contents

List of Figures

List of Tables

Abstract

A solid state electronic two-tap weight linear adaptive neuron has been designed and breadboarded. The electronic neuron employs a novel nonvolatile memory device, configured as an electrically reprogrammable analog conductance, as the synaptic element. A learning rule, known as the Widrow-Hoff's delta rule, is also incorporated in the neuron. Furthermore, the neuron trains itself according to a so-called Clipped-data Least Mean Square Error (CLMSE) learning algorithm implemented in the linear adaptive neuron.

The electrical properties of the nonvolatile memory device for the realization of the analog reprogrammable conductance are examined. These properties include the erase/write, memory retention, and dynamic range characteristics. We have concluded from our investigation the nonvolatile memory device possesses many attractive features which make it an ideal candidate for the implementation of the synaptic element.

The electrical performance of the linear adaptive neuron has been evaluated and presented. Theoretical analysis and a software simulation routine are also included in this thesis. The unique combination of the learning algorithm and the synaptic element serves as an opportunity to study the basic functional block in a large neural network.

Chapter 1
Introduction

1.1 Background

Neural networks have received tremendous attention over the past few years. Parallel architecture, a distinct feature of the neural networks, offers engineers an additional dimension to realize learning machines over the existing serial computers via software simulation. A neural network is a system composed of many simple processing elements in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes. Neural networks can perform high-level tasks such as adaptation, or low-level tasks such as speech recognition, preprocessing sensory input data for vision tasks.[1]

In neuro-biology language, adaptation requires modification of the synaptic strength between neurons. A neuron will fire an electrical impulse when the summation of the all the input signals coupled through their synaptic strengths exceed a certain threshold value. Figure 1-1 shows how a neuron might look like in a biological neural system. In the electrical implementation of neural networks, a neuron cell body is represented by a summing amplifier. The variable synaptic weights are represented by variable resistors. The neural networks also need learning algorithm implementations for updating or adjusting the synaptic weights. Figure 1-2 show a conceptual diagram of an electrical neural network implementation. Figure 1-3 shows a comparison between the biological neuron and its electrical counterpart. A more detailed discussion of how the electrical neuron mimics the biological neuron in each category will be presented in the thesis.

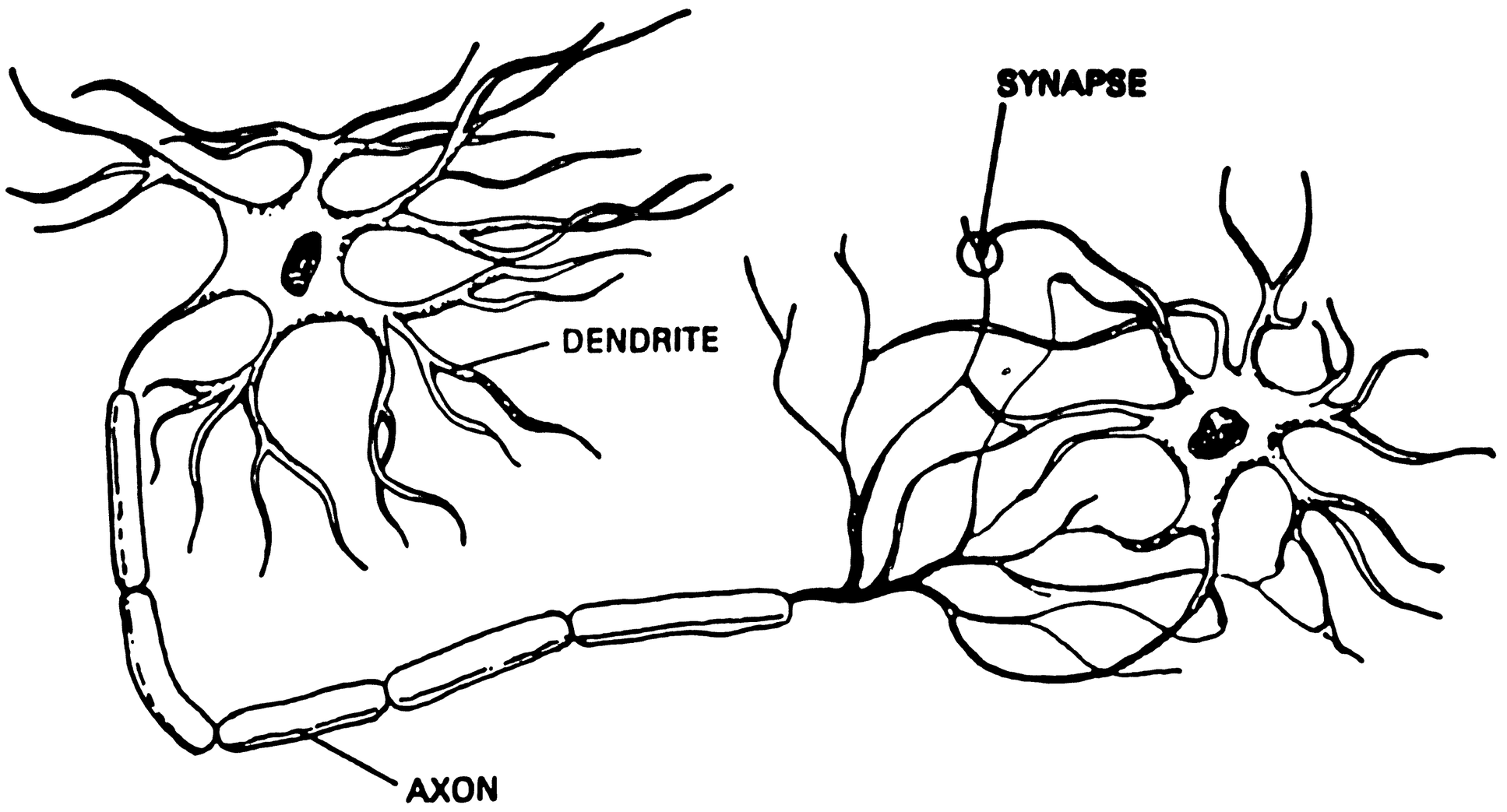


Figure 1-1: Biological Neuron[1]

1.2 Electrical Implementation of Synaptic Weights

1.2.1 Analog versus Digital Hardware Implementation

From a biological point of view, neurons are almost always nonlinear, typically analog in nature and may be slow compared to modern digital circuitry. Neurons may also include temporal integration and other types of time dependencies and also mathematical operations more complex than summation.[1] The question of whether the electrical artificial neural networks should be implemented in the analog fashion or in digital fashion is an on-going

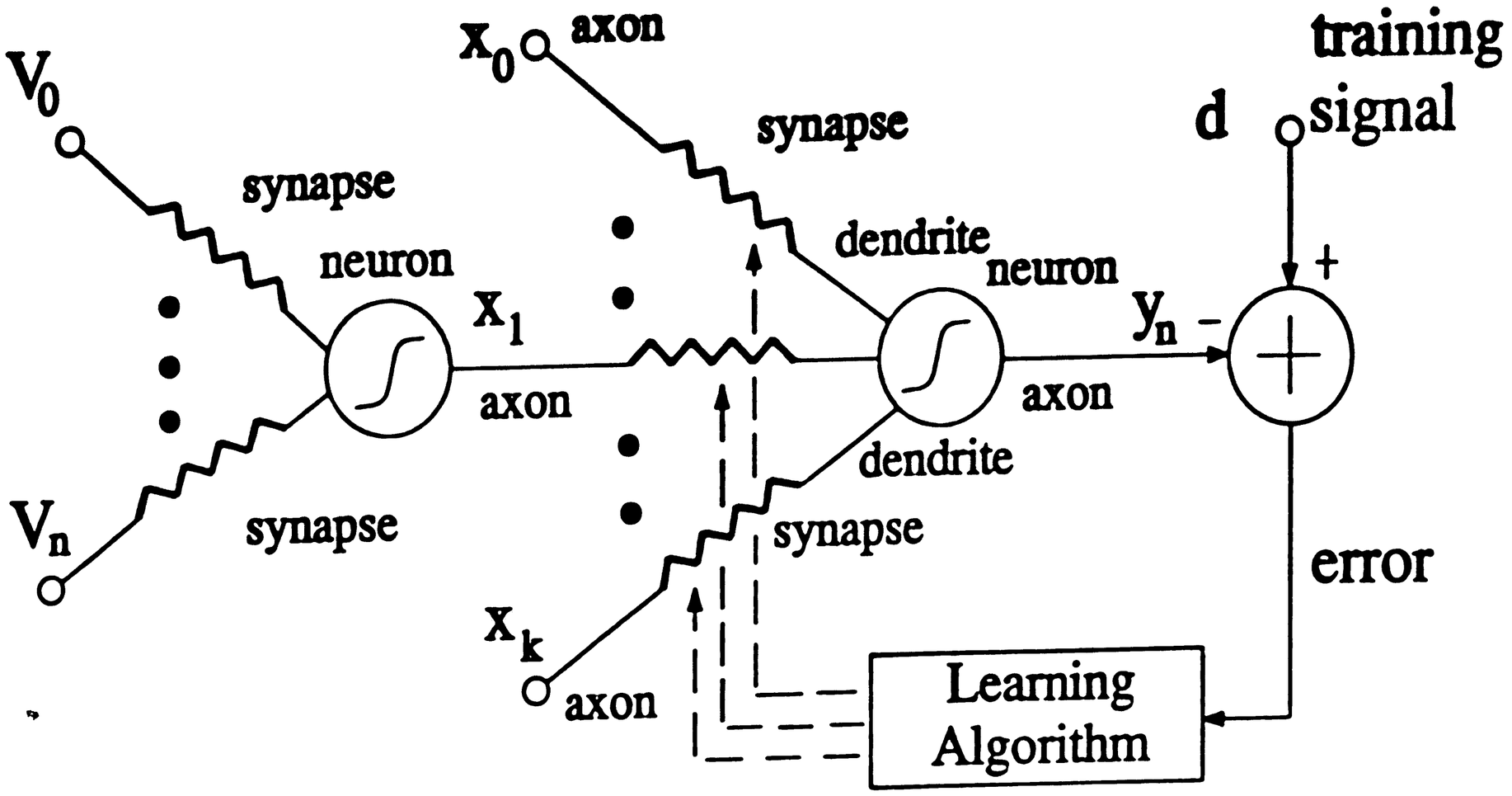


Figure 1-2: Electrical Implementation of Neural Networks

debate. Some researchers have implemented the synaptic weights and the network itself digitally[2]. Digital implementation of the electrical neural networks enjoys the advantages such as the ease of the design because of the vast number of available digital component libraries, the speed of modern digital circuitry, and better immunity to random noise present in the system. However, it also suffers from such drawbacks as, (1) the complexity of the system increases sharply as the designer increases the accuracy of each synaptic weight value (bits/weight), (2) the area consumption of the system increases as the number of the synaptic weight interconnection increases, and (3) the power

Biological Neuron	Electrical Neuron
. Dendrite	Input Line
. Axon	Output Line
. Cell Body	Summing Operational Amplifier
. Synapses	SONOS Nonvolatile Transistors
. Biological Clock	Synchronized Clock
. Nonlinear	Linear Approximation
. Excitatory/Inhibitory Synapse	Positive/Negative Differential Conductance
. Reinforced Learning	Reinforced Reading
. Memory Retention/Loss	Weight Information Retention/Loss

Figure 1-3: Comparison between the Biological and Electrical Neuron

consumption will be a bottleneck for large digital neural networks. Furthermore, the digital approach suffers from the quantization error in the weight value. On the other hand, analog artificial neural networks, although suffering from problems in design and speed, offer attractive features such as lower power consumption, less complexity for large network, small chip area consumption per weight and the absense of the quantization error in synaptic weight values. Therefore, if we seek an efficient hardware implementation of the neural networks, then the network as well as the synaptic weights should be analog. The efforts to implement the synaptic weights with temporary charge storage on the input capacitor of a MOS transistor to alter the analog conductance of the MOS transistor have been reported in the literature.[3,4] This approach gains an advantage in chip area consumption, but the weight is

temporary and requires periodic refresh similar to a DRAM. Therefore, neither approach is suitable for synaptic weight implementation.

1.2.2 Nonvolatile Memory Devices in Neural Networks

A nonvolatile memory cell, on the other hand, provides the features such as read enhancement and retention loss which mimic the biological synapse. Two basic types of nonvolatile memory exist, the floating gate device and the floating trap device. The floating gate device stores weight information in the form of charges on a 'floating' polysilicon layer under a polysilicon control gate. Holler *et. al.*[5] have investigated the implementation of modifiable synaptic weights by using the floating gate devices. Although the floating gate device shares the same desired features with the floating trap device, its high programming voltage become a barrier to overcome in order to realize large electrical neural networks. The conventional floating trap device, known as the Metal Nitride Oxide Silicon (MNOS) memory transistor, had been used as the modifiable synaptic weight previously.[6] The scaled floating trap device, such as Silicon Oxide Nitride Oxide Silicon (SONOS) device, provides salient features of low programming voltage, low power dissipation, wide dynamic range of analog conductance, and small chip area consumption.[7] All these attractive features suggest that SONOS transistor cell should be used in future electrical implementation of the neural networks.

1.3 Scope of this Thesis

My research was initiated to demonstrate the SONOS nonvolatile memory transistors can be effectively used as an electrically modifiable synaptic weights in neural networks implementation. To serve the above-mentioned motivation, a two-tap weight linear adaptive neuron has been designed and breadboarded. I have selected a particular learning algorithm for the linear adaptive neuron,

namely, the clipped-data least mean square (CLMS) error algorithm. The research is intended to provide a means to examine a basic functional neural building block for neural networks.

The work present in this thesis includes an exploration of the operation theory of the two-tap weight linear adaptive neuron, and a description of the available programming algorithms. This thesis also devotes a chapter to the technology of the electrically modifiable synapses, namely the SONOS nonvolatile memory transistors. In addition, I give a detail discussion of the experimental setup and the operation of the two-tap weight linear adaptive neuron.

This thesis identifies the attractive features of the SONOS nonvolatile memory transistors as the electrically reprogrammable synaptic elements for the hardware implementation of the neural networks. Furthermore, the feasibility of the integration of the CMOS VLSI technology with SONOS technology is demonstrated in the thesis. This integration of both technologies is essential for the realization of large neural networks in the future.

Chapter 2
Theory of the Linear Adaptive Neuron

2.1 Discussion of Terminology

The biological neural system is made up of billions of nerve cells, called neurons, massively interconnected to each other to perform all the everyday cognitive activities. Each neuron is composed of three main parts, namely the cell body, the input branches called dendrites, and the output branch called axon as previously shown in figure 1-1. The input and output signal transmission are done in the form of electrical impulses. The cell body acts like a summing point for all the dendrite inputs and produces the output for transmission by the axon. The basic computing nodes sums N weighted inputs and passed the result through a nonlinearity as mentioned before. Figure 2-1 shows three common types of nonlinearities; hard limiter, threshold logic elements, and sigmoidal functions.[8]

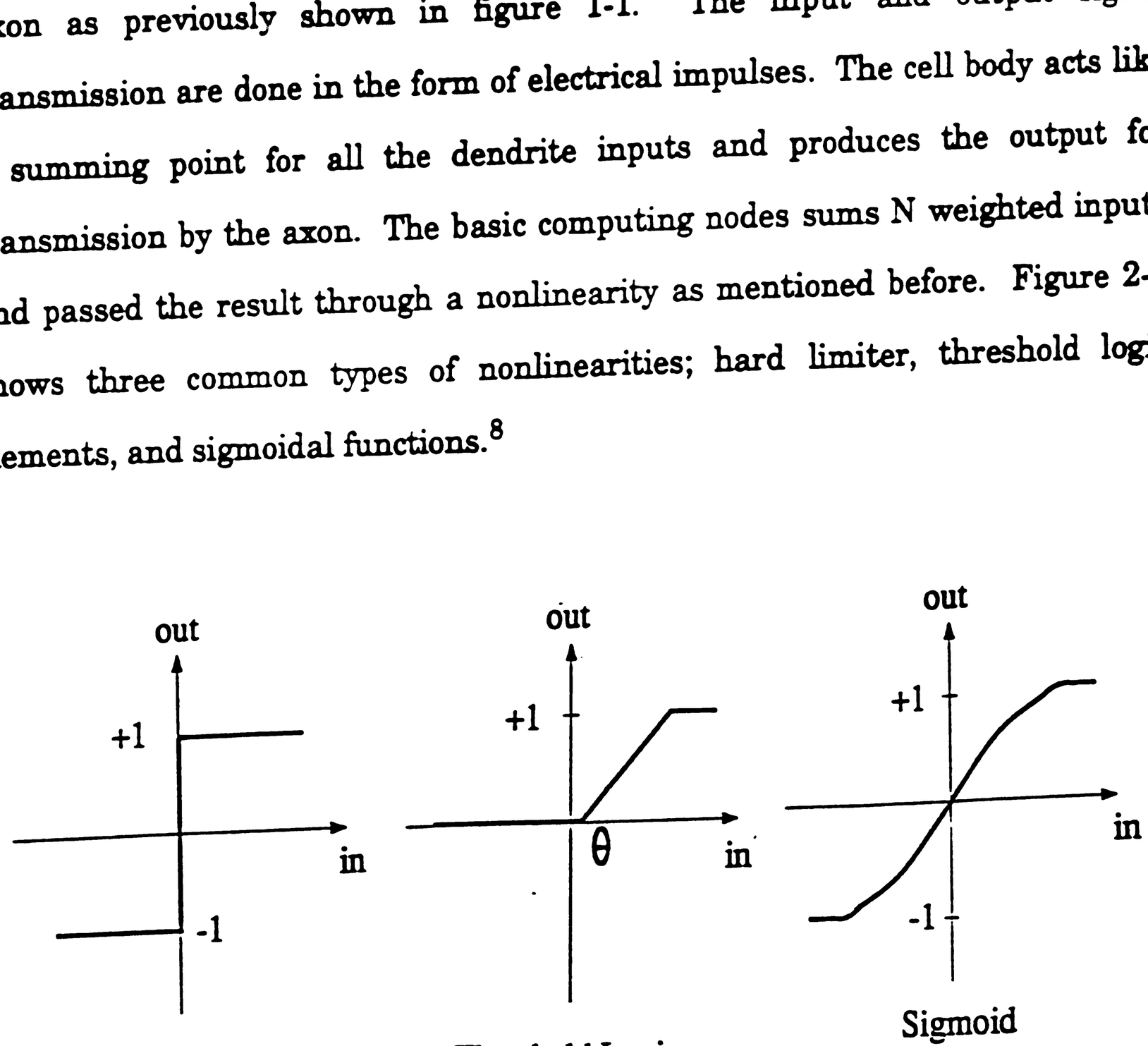


Figure 2-1: Different Type of Nonlinearities in Neuron

The entire neural system is massively coupled, meaning that the dendrites of a particular neuron can be crossovered by the axons from all the neighboring neurons. Sometimes, the axon of a particular neuron can even crossover the dendrite from its own neuron. Wherever the axon and dendrite cross each other, a synaptic interconnection is formed between them. The electrical impulse travelling along the dendrites and axons are caused by the potential difference of the sodium and potassium ions on each side of the membrane which separates the dendrite and the axons from the surrounding fluid. The signal from axon is transferred to the dendrite at these synaptic interconnection by a neurotransmitter. The learning process requires the modification of how the signals are carried or coupled from the axons to the dendrites. The cell body determines whether it is going to fire an electrical impulse or not depending on the result of the summation process. If the result of the summation exceeds a threshold value, the neuron will fire an electrical impulse. Conversely, if the result of the summation is lower than the threshold value, the neuron will not fire any electrical impulses down the axon. Generally speaking, the threshold value is a nonlinear function of the input signal strength. The strength of the signal travelling along a dendrite is determined by the strength of the synapses that connected to the particular dendrite. Two types of synaptic connections have been found in the human body, namely the excitatory and the inhibitory synaptic connection. The excitatory synaptic connection adds the signal to the summation in the cell body while the inhibitory synaptic connection reduces the summation in the cell body.

In order for artificial electrical neural networks to successfully mimic the biological neural networks, the artificial electrical neural networks must have equivalent counterparts to their biological partner. In artificial analog electrical neural networks, an operational amplifier replaces the position of the cell body.

The operational amplifier is configured to serve two purposes, one is to sum up all the current entering in the inverting or noninverting terminal and the other is to convert the summed current into a corresponding voltage for further processing. The output of the operational amplifier represents the axon in the biological neuron. The synapses in the biological neural system are to be replaced by variable analog conductances. Since the inputs of the electrical neural system, acted like dendrites in the biological neural system, are subjected to voltages, the varying analog conductances convert the voltage inputs into currents which are summed by the operational amplifier. In order to achieve programmability, the electrical neural networks must operate with a learning algorithm. The purpose of the learning algorithm is to change the system characteristics such that the output signal matches the teacher signal. Figure 2-2 shows a simplifed block diagram of the linear adaptive neuron. The learning algorithm addressed in this thesis is called the clipped data least mean square error (CLMSE) algorithm. Compared to the more familiar least mean square error (LMSE) algorithm, the CLMSE algorithm clips the amplitude information of the input variable. A more detail discussion on the learning algorithms will be found later in this thesis.

2.2 Architectures of Various Neural Networks

2.2.1 Structure of Neural Networks

The biological neural system consists of numerous neurons interconnected to each other. It is known that not all the neurons and nerve cells look alike. The muscle nerve cell, for example, is long and thin in shape and is responsible for muscle contraction. Most researchers agree that the biological neural system is made up of different 'layers'. Each layer is composed of many neurons serving similar functions. The neurons in the same layers are not only massive

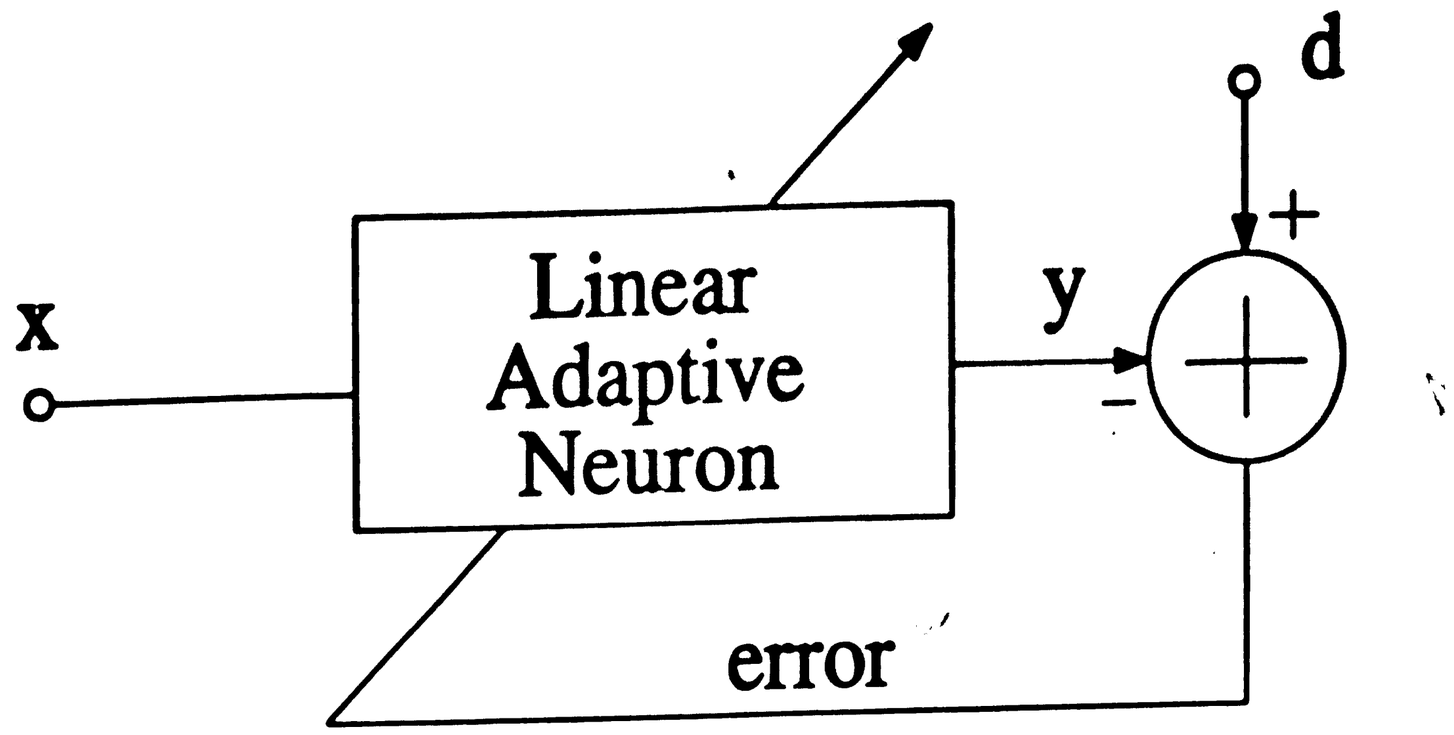


Figure 2-2: Simplified Block Diagram of the Linear Adaptive Neuron

interconnected to each other but also have connections between neighboring layers. A conceptual multi-layer architecture of neural network is shown in figure 2-3.

The first layer is normally the input layer, which can be considered as the sensory neurons in human body. This layer of neurons gathers all the input data and pass it down to the next layer, the hidden layer, which resides between the input and the output layer of the neural system. The hidden layer is responsible for gathering the input information from the input layer and processing the information before sending the result to the output layer or another hidden layer. While the neurons in the input layer may not have interconnections between them, the neurons in the hidden layer are believed to be highly interconnected. The neurons in the output layer sum up the results from the hidden layer and determine their states from the results they receive. This thesis concentrates on the part that consists of two hidden layer neurons and one output layer neuron and the work is highlighted in figure 2-4.

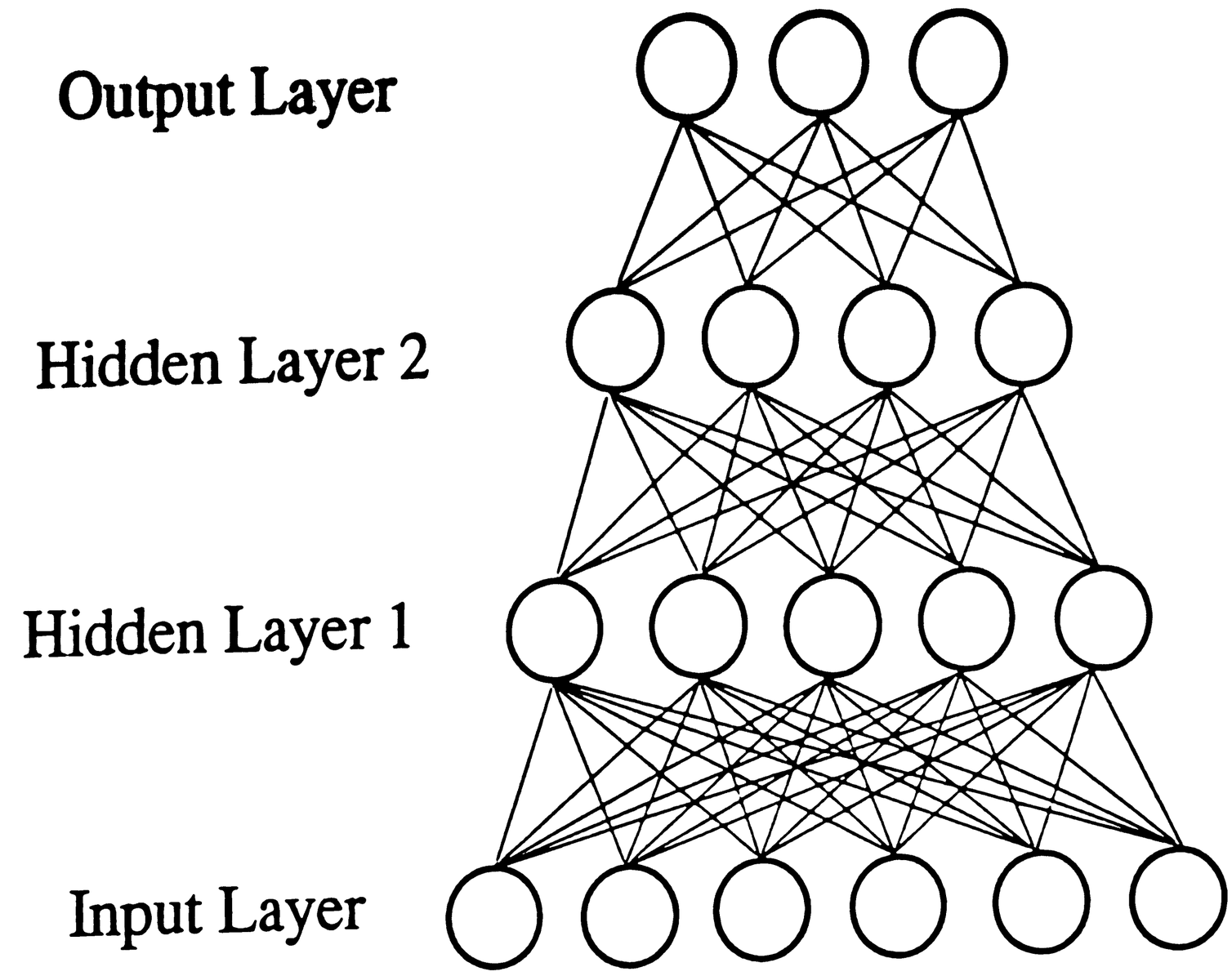


Figure 2-3: Conceptual View of the Multi-layer Neural System

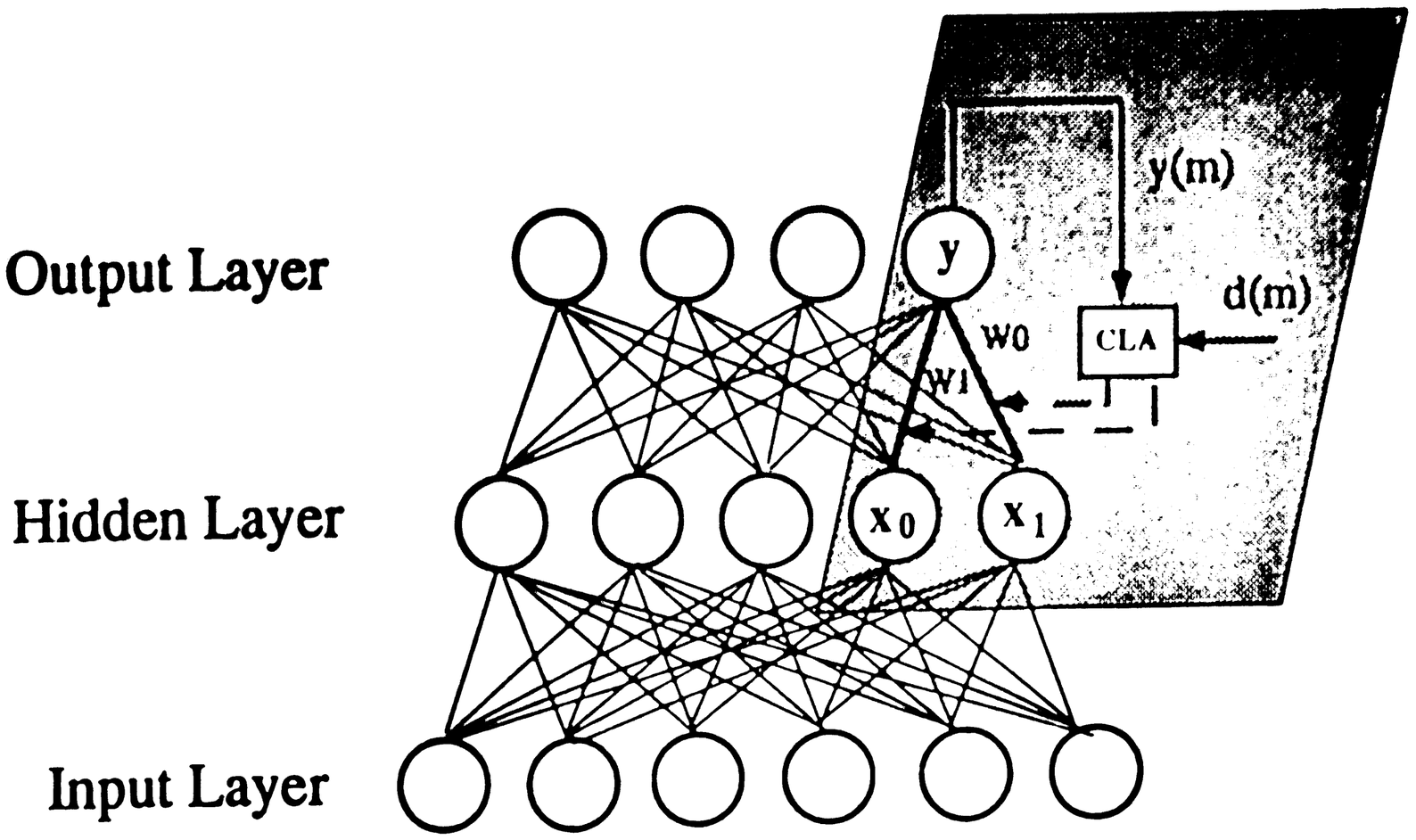


Figure 2-4: Highlight of the Thesis in the Multi-layer Neural System

2.2.2 Classification of Neural Networks

According to the method that the neural networks train or program themselves, the electrical neural networks can be classified into three main categories: Supervised Training, Unsupervised Training, and Self-Supervised Training. The supervised training neural network requires the presence of an external training signal or teacher and the labelling of the data used to train the network. The teacher knows the correct response desired from the network and gives a corresponding error signal when the network produces an incorrect

response. The network then utilizes the error to learn and correct its response. Therefore, this type of neural network requires two input signals, one for system input signals and the other one for the teacher or training signals. Unsupervised neural network, on the other hand, uses unlabeled training data and thus requires no external teacher. When data is presented to the network, it forms internal clusters that compress the input data into classification categories. The third type of neural network, namely the self-supervised training neural network, monitors its performance internally. Therefore, it does not require an external teacher. A corresponding error signal is generated just like its supervised training counterpart, however, this error signal is generated by the system itself. The error signal is also fed back to the system and the correct response is produced after number of iterations. A general review of various classes of the neural network is included in Appendix A. The linear adaptive neuron presented in this thesis belongs to a class of single level perceptrons, which performs supervised training.

2.2.3 Topology of Neural Networks

The actual implementation of the interconnections of the biological neural system is still an open question. However, there are a limited number of available electrical interconnections for the implementation of neural networks. Currently, three types of the interconnection implementation across the same layer are under investigation. They are: the locally connected network, the fully connected network, and the sparsely connected network. In the locally connected network, the interconnections are made between the neighboring neurons only. In the fully connected network, the interconnections are made from one neuron to all other neurons in the same layer. Where the sparse connected system is applicable, the interconnections are created from one

neuron to a few distant neurons in the system. There are two types of connection between different layers in the system, namely, the feedforward connection and the feedback connection. In the feedforward connection system, the connections are made only from the lower layer of neurons to the higher layer of neurons. This network only works until the output comes out from the highest layer. An example of this type of connection are the connections between the neurons in the input (sensory) layer of the neural network to the neurons in the hidden layer. On the other hand, in the feedback connection system, part of the higher layer output is fed back as the inputs for the lower layer. This type of operation depends on the iteration process of the system to produce the final output. An example of this type of connection are the connections between the output layer neurons and the hidden layer neurons in the system. It should be pointed out that this thesis addresses the electrical implementation of a neural network that operates on the principle of supervised training, utilizing feedback (backpropagation) connection between the input and output layer.

2.3 Advantages and the Applications of Neural Networks

Electrical neural networks have many advantages over the traditional serial computer. First of all, the neural networks are naturally parallel, while the von Neumann architecture, on which modern computers are built, is inherently serial in nature. Neural networks find rules for Large Complex Systems (LCS) rather than follows models implemented in serial computers that might be too simple or rigid or too detailed and complicated for the time scales involved.[9] In addition, neural networks are adaptive or trainable. Artificial Intelligence (AI), implemented on the von Neumann computers, has to depend on the preset rules or models in order to learn or adapt. In other words, AI has

to operate in the environment which is knowable and controllable. On the contrary, the adaptive learning in the neural networks can operate in an unknown or noisy environment. Due to the massive parallelism, neural networks have a lot more fault tolerance than serial computers. If part of the neurons in the neural networks are destroyed, then the collective effort of the network will still produce a correct response. On the other hand, if part of the AI software code is destroyed, then the output of the serial computer may not produce the desirable response. Since all electrical components deteriorate with time, the serial computers performance will become worse as time progress, however, the parallelism of the neural networks circumvents this problem since the output is determined by the collective efforts of all the components, not just one or a few components. In addition, the neural networks are insensitive to small variations between computing elements. One of the most important advantages of a neural network is its ability to perform tasks in the real time domain. If von Neumann computers are used to simulate the real time performance of neural networks, then they must have tremendous computing speed and memory space, which makes the system unreliable. Neural networks provide an alternative way for real time system control and real time signal processing areas, which are gaining a lot more attention over the years. Figure 2-5 shows a schematic comparison between AI and Neural Networks.

The emergence of neural networks has promised several applications for which their serial computer counterpart is inefficient in performing. Pattern recognition, a computation intensive job, is one of the natural applications for neural networks. If the task of pattern recognition is to be performed by the serial computer, then the computation and memory requirements for this task will force these computers to process in a non-real time environment. This bottleneck must be solved if the task requires real time control and signal

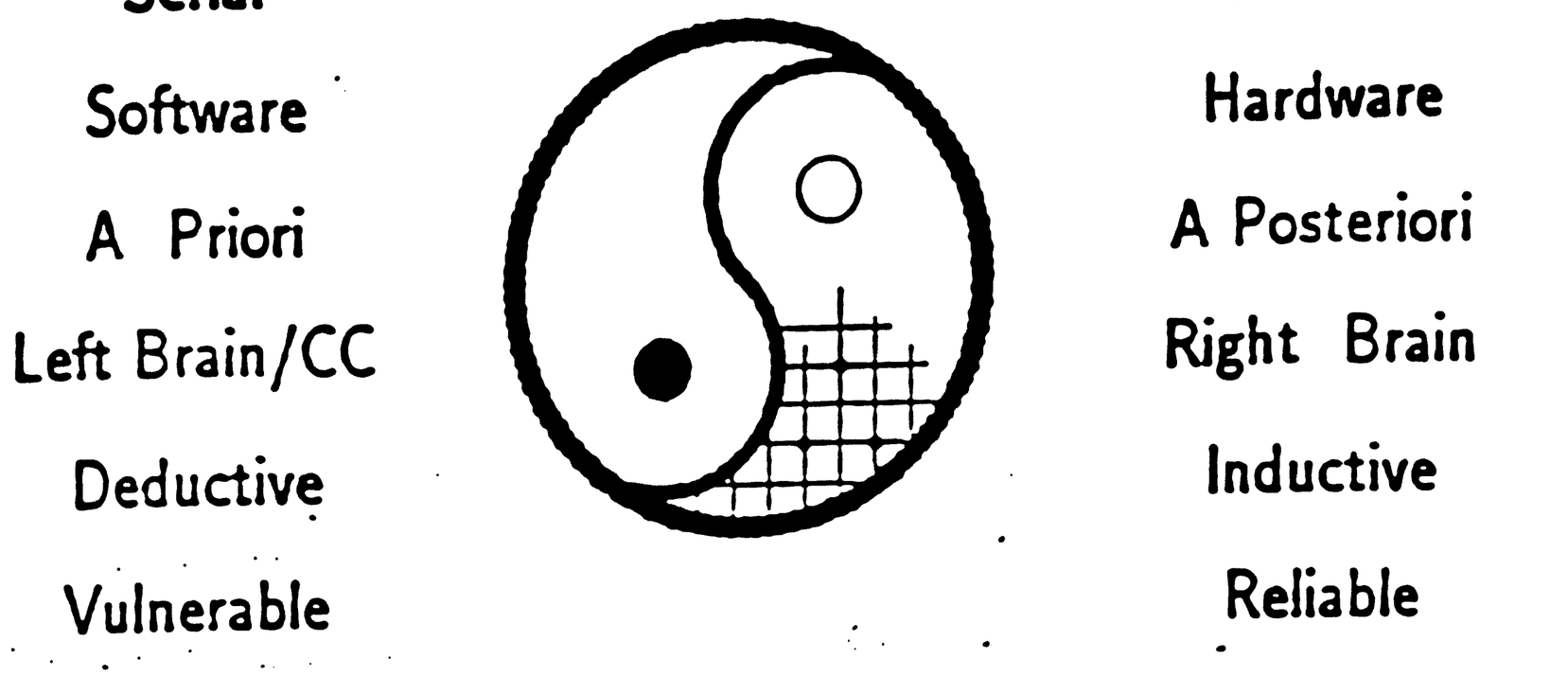


Figure 2-5: Schematic Comparison between AI and Neural Networks[9]

processing. One the other hand, the inborn massive parallel computation of neural networks can reduce the work load of individual computing elements, and thus, real time processing can be achieved. Other applications of neural networks include associative memory. In a conventional approach, each stored word is retrieved by providing its address, where in the associative memory application, a memory word is retrieved by providing part of the word itself. If the given example is a reasonable match to the corresponding part of the stored word, the entire corrected word will appear at the memory output.[10] The data compression and speech prediction capabilities of neural networks make them an excellent candidate in the area of speech communications. Therefore,

communication systems need only transmit vital information/data to the receiving end where the entire speech can be reconstructed by neural networks. One of the applications for neural networks is to configurate the neural network as an adaptive signal processor for system modeling. Once the adaptive signal processor converges, the transfer function of the neural network represents the transfer function of the unknown system. Recently, parallel distribution processing has provided another territory for neural networks. As time progresses, neural networks have found their usefulness in fields where traditional techniques are determined to be ineffective. In the future, neural networks may integrate with modern signal processing techniques to attain optimum system performance.

2.4 Synaptic Weight Formulation

The synaptic weights in electrical neural networks require programmability of the analog conductances. The programmable analog conductance in our work utilizes the variable threshold voltage of the nonvolatile memory devices, such as an EEPROM. If we restrict our operation of the nonvolatile memory transistor to the linear region, then the drain current of the transistor can be written as,

$$I_d = \beta_{eff}\{(V_{gs}-V_{th})\cdot V_{ds} - \frac{(V_{ds})^2}{2}\} \tag{2.1}$$

where

$$\beta_{eff} = \mu_{eff}(W/L)C_{eff} \tag{2.2}$$

μ_{eff} is the effective electron mobility, C_{eff} is the effective gate dielectric

capacitance per unit area, and W/L is the transistor geometry width to length ratio. The channel conductance of the transistor can be expressed as,

$$g_{ds} = \frac{\partial I_{ds}}{\partial V_{gs}} \tag{2.3}$$

$$= \beta_{eff}\left[(V_{gs} - V_{th}) - (V_{ds})\right]$$

where V_{th} is the electrically modifiable threshold voltage of the nonvolatile memory devices. There are two ways to alter the analog channel conductance of a semiconductor transistor: (1) fix the threshold voltage but vary the gate to source voltage of a transistor, or (2) fix the gate to source voltage but vary the threshold voltage of a transistor. A conventional Metal Oxide Semiconductor (MOS) transistor offers a fixed threshold voltage. Therefore, the programming of the analog conductance of a MOS transistor depends on the charging and the discharging of the gate capacitor. However, the charges on the gate capacitor slowly leak out due to the leakage currentof the address switch, resulting a loss in weight information. Therefore, some form of additional control/refresh circuitry is required to maintain proper storage of the weight information. (i.e. this approach lacks the nonvolatile property of the neuron). In our work, we take the second approach. We program the analog conductance of the weight element by programming the threshold voltage of the nonvolatile memory transistor. The nonlinear term V_{ds} in equation (2.3) can be ignored if either (1) the value of V_{ds} is kept small or (2) the neural network is implemented in such a way so that the nonlinear term in the conductance is removed after the summation operation. In our work, the latter approach is employed.

The electronic synapses must be bipolar in weight value, that is, they have to be either positive or negative at any given time. The positive weight

values represents the excitatory synapses and the negative weight values represents the inhibitory synapses. Since one single transistor can only be positive in conductance value, we have taken the approach of using two analog programmable conductances which share a common drain voltage as one single weight element. Those two analog conductances are configured in such a way that the actual weight value is proportional to the difference in analog conductances. If we denote one of the analog conductance in the weight element to be g_{ds}^{+} and the other g_{ds}^{-}, then we can write the weight as below:

$$\begin{aligned} W &\alpha\ g_{ds}^{+} - g_{ds}^{-} \\ &\alpha\ \beta_{eff} \cdot [(V_{gs} - V_{th}^{+}) - (V_{gs} - V_{th}^{-})] \\ &\alpha\ \beta_{eff} \cdot [V_{th}^{-} - V_{th}^{+}] \end{aligned} \tag{2.4}$$

where V_{th}^{+} and V_{th}^{-} are the threshold voltages associated with the analog conductances of g_{ds}^{+} and g_{ds}^{-}. Therefore, if the value of V_{th}^{+} is smaller than the value of V_{th}^{-}, then the weight is positive in value. Conversely, if the value of V_{th}^{+} is larger than the value of V_{th}^{-}, then the weight will have negative value.

The error generated in the linear adaptive neuron is used to formulate the weight increment (or decrement if the error is negative). In our work, we employ the Widrow-Hoff's Delta Rule[11] for the weight update algorithm. The Delta Rule can be expressed as follow,

$$W_{new} = W_{old} + \Delta W \tag{2.5}$$

where the ΔW is the incremental weight. Using Equation (2.4), the incremental change in the weight value is proportional to an incremental change in the differential threshold voltages. Since the incremental weight value varies

proportionally with the error, the quantity ΔW will have its largest value when the linear adaptive neuron is in the early stage of adaptation where the error is the largest. The incremental change in the weight value will decrease in value as time progress as the neuron adapts to its desired response. If the incremental weight value is too large, then the linear adaptive neuron may overcorrect itself. Under this condition, the error sign will change and steer the neuron toward the desired output.

2.5 The Programming Algorithm

The operation of the linear adaptive neuron is based on the linear combiner with a learning algorithm which is responsible to reconfigurate the elements in the linear combiner to obtain desired response. Figure 2-6 shows a schematical diagram of a linear combiner. The input signal is fed into a delay line and tapped out nondestructively. The tapped signals are then multiply with their respective weights and linearly combined at the summer. The output of the summer represents the neuron output and is compared to a desired signal which the output of the neuron is trained to match. The linear adaptive neuron compares the output of the neuron to the desired signal in order to determine how the weight elements should be altered.

In our work, the input tapped signals are $x_0(m)$ and $x_1(m)$, and their prospective synaptic weight interconnections are $\dot{W_0}(m)$ and $W_1(m)$. The output of the neuron can be expressed as,

$$y(m) = \sum_{k=0}^{1} W_k(m) \cdot x_{m-k} \tag{2.6}$$

where m is the time index and k is the spatial tap index. Equation (2.6) can be expressed in a matrix form. The input signal vector is defined as,

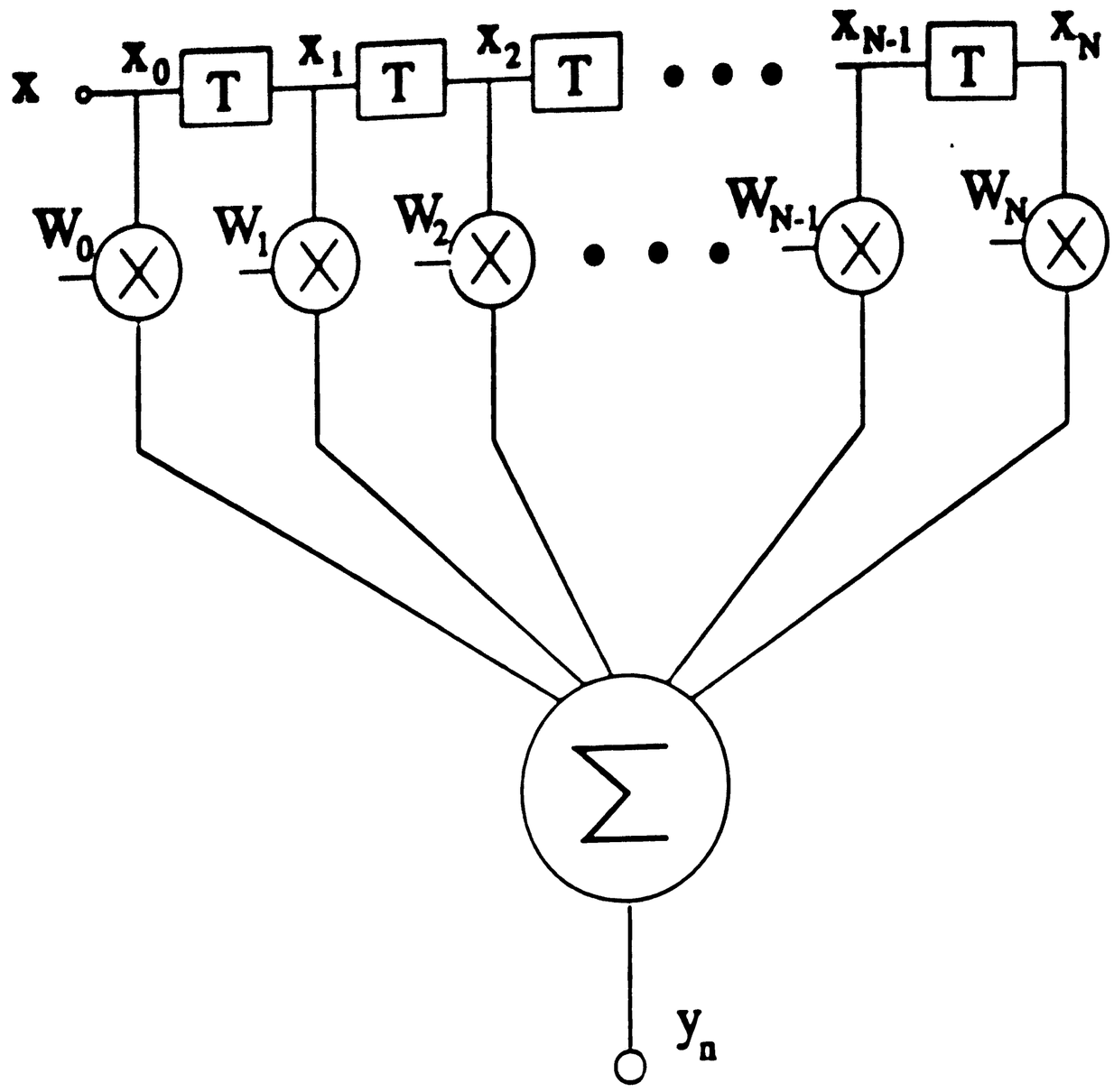


Figure 2-6: Schematic Diagram of the Linear Combiner

$$x_m \equiv \begin{pmatrix} x_0(m) \\ x_1(m) \end{pmatrix} = \begin{pmatrix} x_{m-0} \\ x_{m-1} \end{pmatrix} \tag{2.7}$$

and the adjustable weight vector is defined as,

$$W \equiv \begin{pmatrix} W_0(m) \\ W_1(m) \end{pmatrix} \tag{2.8}$$

then we can express the output y_m in the matrix form as,

$$y_m = W^T \cdot x_m \tag{2.9}$$

where W^T is the transpose of the weight vector. An error is generated when there is a mismatch between the output of the neuron y_m and the desired response d_m for the neuron. The error generated is then defined as follow,

$$\begin{aligned} \varepsilon_m &= d_m - y_m \\ &= d_m - W^T \cdot x_m \\ &= d_m - x_m{}^T \cdot W \end{aligned} \tag{2.10}$$

The purpose of the learning algorithm is adjust the weights in the system in order to minimize the error generated. Two types of learning algorithms will be discussed in the following sections are the Least Mean Square Error (LMSE) algorithm and the Clipped-data Least Mean Square Error (CLMSE) algorithm. Both algorithms require the minimization of the mean or the average of the square of the error generated. The squared error can be written as,

$$\varepsilon_m{}^2 = d_m{}^2 - 2dx_m{}^T W + W^T x_m x_m{}^T W \tag{2.11}$$

If we want to take the mean of a quantity, we have to take the expected value of that quantity. Therefore, we can express the expected value of the square of error generated as,

$$E(\varepsilon_m^2) = E(d_m^2) - 2P^T W + W^T R W \tag{2.12}$$

where P is the cross-correlation vector between the desired response and the input signal and R is the input correlation matrix defined as,

$$P \equiv E\begin{pmatrix} d_m x_m \\ d_m x_{m-1} \end{pmatrix} \tag{2.13}$$

$$R \equiv E\begin{pmatrix} x_m x_m & x_m x_{m-1} \\ x_{m-1} x_m & x_{m-1} x_{m-1} \end{pmatrix} \tag{2.14}$$

Equation (2.12) is a quadratic function of the weight. The process of weight adjustment, in order to minimize the mean squared error is to take the gradient of equation (2.12) with respect to the weight.

$$\nabla_W E(\varepsilon_m^2) = -2P + 2RW \tag{2.15}$$

The optimum weight vector, generally called the Wiener weight vector, is obtained by setting the gradient equal to zero and to yield

$$W_{opt.} = R^{-1} P \tag{2.16}$$

The minimum square error is then obtained by substituting equation (2.16) into equation (2.12).

$$\{ E(\varepsilon_m^2) \}|_{min.} = E(d_m^2)| - P^T W_{opt.} \tag{2.17}$$

It is obvious that the minimization process requires intensive computational operations such as summation, averaging and matrix multiplication. This becomes a bottleneck which needs to be removed when the system increases in size. We will employ a reasonable approximation for the squared error, called the Widrow-Hoff Mean Square Error Algorithm, which will be discussed in the next two sections.

2.5.1 The Least Mean Square Error Algorithm

The least mean square algorithm utilizes the method of steepest descent to minimize the difference between the desired signal and the output of the neuron. Therefore, the weight adjustment is in the direction proportional to the negative gradient of some function F, where F is the mean square error or the expected value of the square of the error signal. Therefore, we can express the weight adjustment as

$$\frac{dW}{dt} = -\mu \nabla_W E(\varepsilon^2) \tag{2.18}$$

$$= -\mu \nabla_W \overline{\varepsilon^2}$$

where μ is called the convergence factor. Since our work implements a discrete-time, sample-data system, we can write the differential equation of equation (2.18) as a difference equation

$$W(m+1) = W(m) - \mu \nabla_W \overline{\varepsilon^2} \tag{2.19}$$

where m is the time index. It is quite obvious that equation (2.19) is in the form of delta rule as shown in equation (2.5). If we substitute the result of equation (2.15) into equation (2.18), then we can write the difference equation in the matrix form as

$$W(m+1) = (I - 2\mu R)W(m) + 2\mu P \tag{2.20}$$

where I is the identity matrix. Notice that equation (2.20) is a first order difference equation which can be solved iteratively for the response of the mean weight vector,[12]

$$W(m) = (I - 2\mu R)^m W_0 + R^{-1}\left[I - (I - 2\mu R)^m\right]P \tag{2.21}$$

If we combine equation (2.19) and (2.10), then we can write the jth element of the difference equation as

$$W_j(m+1) = W_j(m) + 2\mu \overline{\varepsilon(m)x(m-j)} \tag{2.22}$$

The LMS error algorithm shown above does not require explicit measurements or calculations of the correlation matrix and, thus, it does not require large memory space for the purposes of matrix storage and inversion calculation. Since no matrix inversion is necessary in this algorithm, the computation time

can be dramatically reduced. Notice, the incremental weight update is the cross-correlation between the error and the input signal vector. The update action will be stopped if the error and the input signal vector are orthogonal to each other. Therefore, the system will converge if there is no common part between the input and the error signal. This feature is extremely attractive if the system is configured to perform noise and echo cancellation.

A particular form of the LMS error algorithm, introduced by Widrow and Hoff, is the approximation[13]

$$\overline{\varepsilon^2(m)} \approx \varepsilon^2(m) \tag{2.23}$$

which eliminates the need to compute the average shown in equation (2.22). Therefore, we can rewrite equation (2.22) as

$$W_j(m+1) = W_j(m) + 2\mu\varepsilon(m)x(m-j) \tag{2.24}$$

Although the approximation made by Widrow and Hoff greatly simplifies the computation of the incremental weight, it requires the implementation of a four quadrant multiplierat each weight location. We shall investigate another form of the LMS error algorithm, namely, the "clipped-data" LMS error (CLMSE) algorithm discussed in the next section.

2.5.2 The Clipped Data Least Mean Square Error Algorithm

The four quadrant multiplier needed to implement the LMSE algorithm poses a cost and area concern for hardware IC realization. Therefore, the replacement of the four quadrant analog multiplier with a non-ideal multiplier has been proposed[14] to overcome this barrier. Although the non-ideal multiplier has quite different characteristics from the conventional four quadrant multiplier, the implementation of the nonideal multiplier is easy and inexpensive. Thus, the convergence of the system with the nonideal multipliers is an important property to be considered carefully. The general form of a nonideal multiplier can be written as

$$(a \cdot b)(t) = a(t) f[b(t)] \tag{2.25}$$

where $(a \cdot b)$ is the output of the nonideal multiplier with the input signals of a and b. $f[b(t)]$ denotes a monotonic function operating on the input signal $b(t)$. Let us consider a particular type of monotonic function, such as

$$\begin{aligned}(\varepsilon \cdot x)(t) &= \varepsilon(t) f[x(t)] \\ &= \varepsilon(t)\, sgn(x(t))\end{aligned} \tag{2.26}$$

where the input signal is "clipped" to retain just the sign information. Moschner had referred to the algorithm

$$W_j(m+1) = W_j(m) + 2\mu\varepsilon(m)\, sgn[x(m-j)] \tag{2.27}$$

as the clipped-data LMS error algorithm. The N multiplications in the LMSE algorithm are then replaced by N conditional branch operations. Depending on

the sign of $x_j(m)$, the incremental weight is either $+2\mu\varepsilon$ or $-2\mu\varepsilon$. Therefore, explicit multiplication of the input data is eliminated. However, this simplification of the LMSE algorithm results in increase in the convergence time by a factor of $\pi/2$ for the case of Gaussian Noise. Equation (2.27) can be rewritten as

$$W_j(m+1) = W_j(m) + 2\mu|\varepsilon(m)|\, sgn[\varepsilon(m)]\, sgn[x(m-j)] \tag{2.28}$$

where $sgn[\varepsilon(m)]\, sgn[x(m-j)]$ represents binary multiplication and, thus, an exclusive OR gate can be used to perform this function. The output of the exclusive OR gate can be used to control a single-pole double-throw (SPDT) switch to steer either the $\pm 2\mu|\varepsilon(m)|$ to increment or decrement the weight. Other forms of the LMSE algorithm are available and will be outlined in the next section.

2.5.3 Other Forms of Least Mean Square Error Algorithm

In the previous sections, the correlation functions of both LMSE and CLMSE algorithms are discussed and we may quantify this aspect by a correlation coefficient defined as

$$\rho_j\,(LMSE) = \sum_{m=1}^{M} \varepsilon(m)\, x(m-j) \tag{2.29}$$

$$\rho_j\,(CLMSE) = \sum_{m=1}^{M} \varepsilon(m)\, sgn[x(m-j)] \tag{2.30}$$

Other algorithms treated in the literature are[15]

$$\rho_j\,(Hybrid) = \sum_{m=1}^{M} sgn[\varepsilon(m)]\,x(m-j) \tag{2.31}$$

$$\rho_j\,(Modified\ Zero\ Forcing) = \sum_{m=1}^{M} sgn[\varepsilon(m)]\,sgn[x(m-j)] \tag{2.32}$$

$$\rho_j\,(Zero\ Forcing) = \sum_{m=1}^{M} sgn[\varepsilon(m)]\,sgn[y(m-j)] \tag{2.33}$$

A comparison of all the various algorithms is shown in figure 2-7 for the equalized peak distortion defined as

$$D_{peak} = \sum_{j=1}^{N} |\rho_j| \tag{2.34}$$

where the summation is valid for all N taps except the center tap.

It is quite obvious that for simpler hardware implementation of the algorithm, the system will suffer from longer convergence time or even instability. Therefore, the trade off between simple hardware implementation and the system performance requires serious consideration.

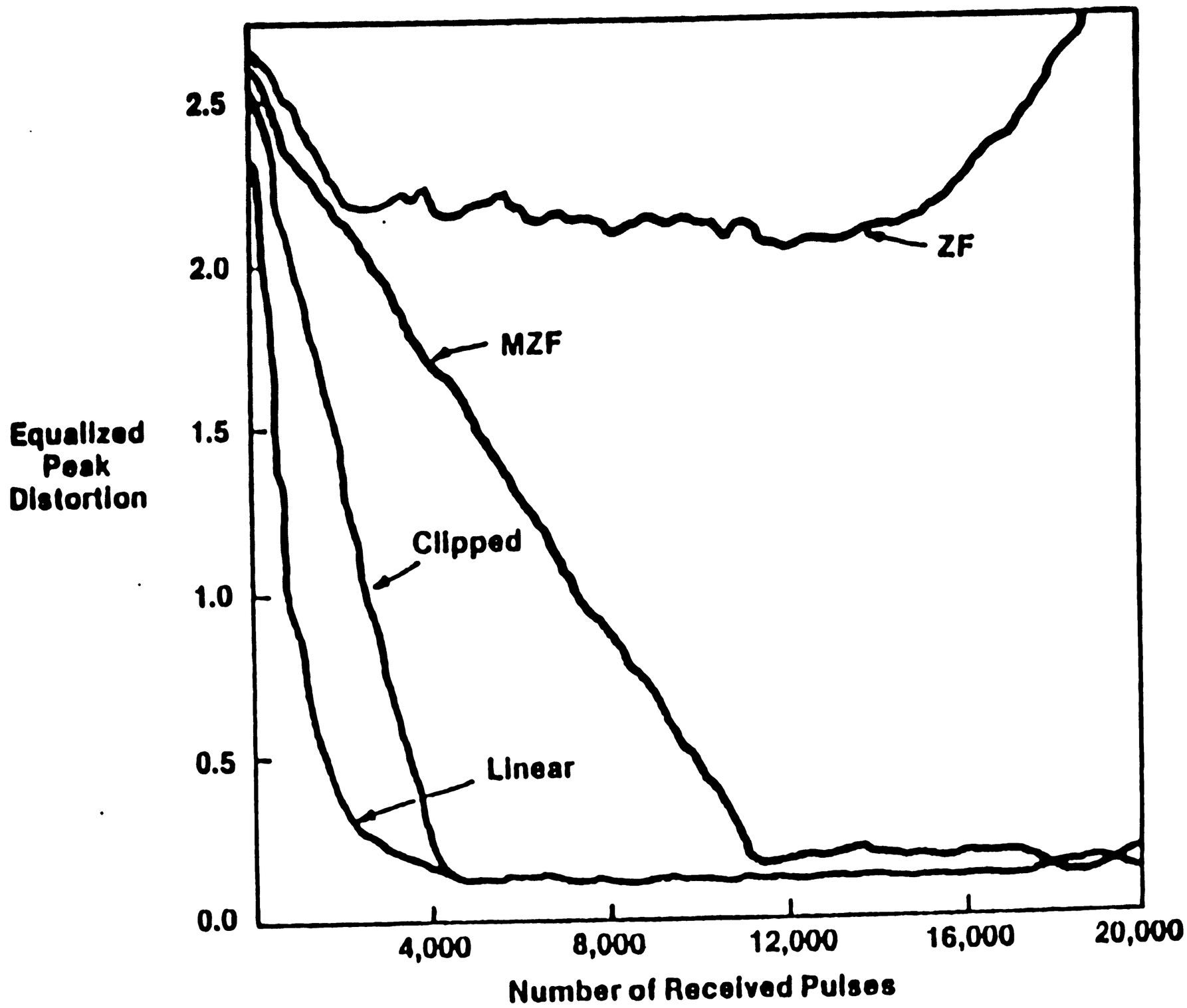


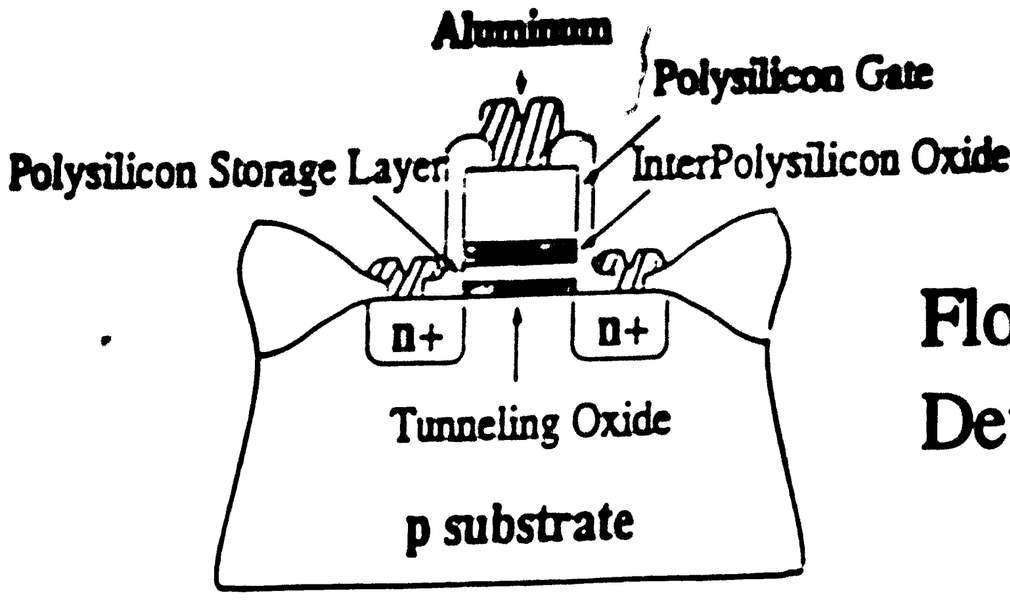
Figure 2-7: Comparison of Various Algorithms for an Adaptive Equalizer[15]

Chapter 3
Technology and Characteristics of Modifiable Synapses

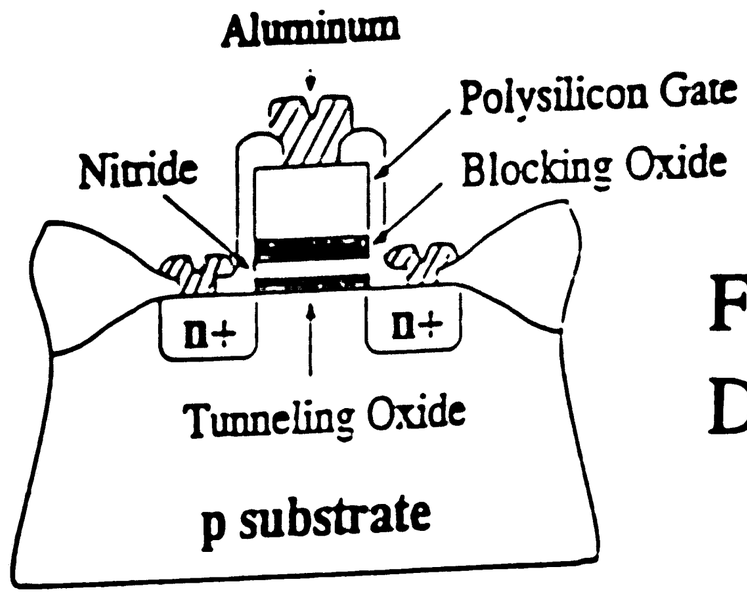
3.1 Background

The analog electrically reprogrammable synaptic weight is composed of nonvolatile memory transistors. Nonvolatile semiconductor memory transistors, unlike regular semiconductor memory such as DRAM or SRAM, retain their memory information even when the power supplies are off. There exist two basic type of nonvolatile memory transistors, namely the floating gate devices and the floating trap devices. Figure 3-1 shows a comparison diagram between these two types of devices. The floating gate devices typically have a fairly thick tunneling oxide (80-100 Å) underneath the polysilicon storage layer and store charges in the polysilicon layer as free charges in the conduction band. On the other hand, the floating trap devices normally have a thinner tunneling oxide (20-30 Å) and the charges are stored in deep level traps in the nitride layer. The analog electrically reprogrammable synaptic weights used in this thesis belong to the floating trap devices.

The earlier version of the nonvolatile semiconductor memory devices was in the form of **M**etal **N**itride **O**xide **S**emiconductor (MNOS) devices. Recent efforts at Lehigh have scaled down the multi-layer gate dielectric dimensions as well as the programming voltages with so-called **S**ilicon Blocking-**O**xide **N**itride Tunneling-**O**xide **S**ilicon (SONOS) devices. Figure 3-2 illustrates a comparison between the MNOS and the SONOS devices. The main difference between the MNOS and the SONOS is the incorporation of an additional layer of blocking oxide underneath the gate electrode. The blocking oxide, typically with the thickness of 35Å - 55Å , is to prevent the injection of the carriers from the gate



Floating Gate Device

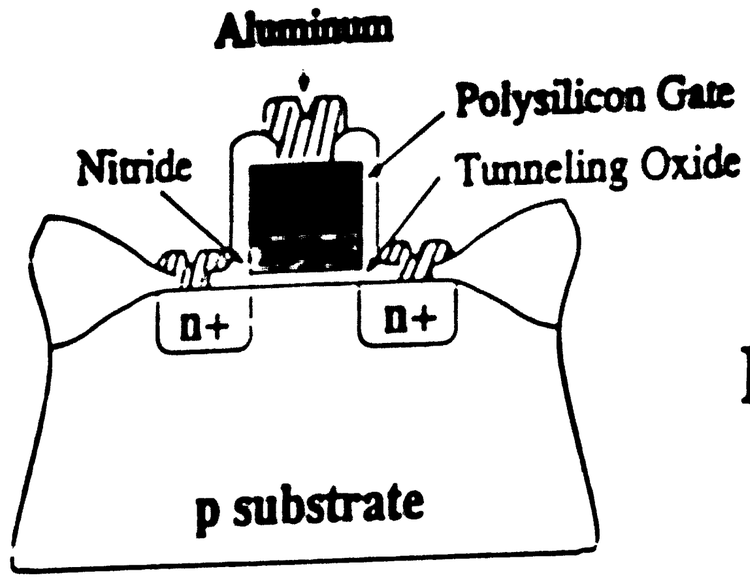


Floating Trap Device

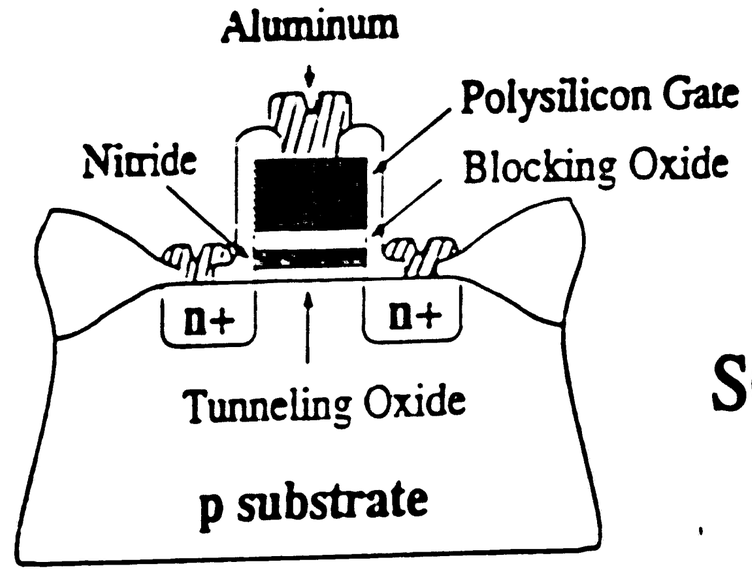
Figure 3-1: Comparison between a Floating Gate Device and a Floating Trap Device

electrode and to prevent the stored charges in the nitride layer from tunneling to the gate electrode.

To operate the SONOS devices, there exist four possible operation conditions: erase, write, read, and idle. The write state is defined as the low conductance state and the erase state is defined as the high conductance state. We can program the SONOS device to either of those two states by a dual power supply or a single power supply, to the four terminals of the devices, namely, the drain, source, gate and bulk. If we use a dual power supply, i.e. either the positive or the negative voltage is available to the gate, then we can summarize the programming operation in table 3-1. V_p is the programming gate voltage to alter the threshold voltage and therefore the analog conductance of the SONOS device, V_r is the reading voltage which is normally chosen to be at the middle of



MNOS



SONOS

Figure 3-2: Comparison between the MNOS and the SONOS device

Operational Mode of a N-channel SONOS Device with a Dual Power Supply				
mode	drain	source	gate	bulk
erase	0	0	$-V_p$	0
write	0	0	$+V_p$	0
read	V_{ds}	0	V_r	0
idle	0	0	0	0

Table 3-1: Operation Mode of SONOS Devices with a Dual Power Supply

two extreme threshold voltages when the device is either in its fully erased state or in its fully written state, and V_{ds} is the drain bias applied during the read operation. In this table, the bulk terminal is tied to the source terminal to avoid

the so called body effect of the transistor during operation. If, however, only single power supply is available for programming, then the operations can be summarized as shown in table 3-2 Throughout the thesis, the programming

Operational Mode of a N-channel SONOS Device with a Single Power Supply				
mode	drain	source	gate	bulk
erase	V_{ds}	V_p	0	V_p
write	0	0	V_p	0
read	V_{ds}	0	V_r	0
idle	0	0	0	0

Table 3-2: Operation Mode of SONOS Deivces with a Single Power Supply

operations of the SONOS synaptic weights are done with a dual power supply. Therefore, the erase operation of the SONOS devices is performed with a negative gate bias; while the write operation of the SONOS devices is performed with a positive gate bias.

If we perform a C-V measurement of the SONOS device, then we can extract the value of the flatband voltage of the device. The flatband voltage of the SONOS device can be expressed as[16]

$$V_{fb} = \Phi_{gs} - \frac{Q_f}{C_{eff}} - \left[\frac{x_{ob}}{\varepsilon_{ox}} + \frac{x_n - \bar{x}}{\varepsilon_n}\right] Q_N \tag{3.1}$$

where Φ_{gs} is the work function difference between the gate material and the bulk semiconductor, Q_f is the fixed charge at the Si-SiO_2 interface, Q_N is the trapped charge in the nitride layer, x_{ob} and x_n are the blocking oxide and nitride thicknesses respectively, $\bar{x}$ is the charge centroid location, ε_{ox} and ε_n are the

permittivities of the oxide and the nitride, and C_{eff} is the effective capacitance of the device given as

$$C_{eff} = \frac{\varepsilon_{ox}}{x_{eff}} \tag{3.2}$$

and

$$x_{eff} = x_{ob} + \frac{\varepsilon_n}{\varepsilon_{ox}} x_n + x_{ot}$$

where x_{ot} is the tunneling oxide thickness. In equation (3.2), we assume the permittivities of the tunneling oxide and the blocking oxide are the same. However, it may not be the case as the tunneling oxide is more silicon rich in nature while the blocking oxide is more oxynitride in nature.

The threshold voltage of the SONOS device can be extracted by normal I-V characteristics of the device. We can write the threshold voltage of the device as

$$V_{th} = V_{fb} + 2\Phi_F + \frac{\sqrt{4\varepsilon_s q N_B \Phi_F}}{C_{eff}} \tag{3.3}$$

$$\Phi_F = \frac{kT}{q} ln\left(\frac{N_B}{n_i}\right) \tag{3.4}$$

where Φ_F is the bulk potential, ε_s is the permittivity of silicon, N_B is the bulk doping density, $\frac{kT}{q}$ is the thermal voltage, and n_i is the intrinsic carrier density. Sometimes, the measurement of the device characteristics yields the so called

"turn-on" voltage. The turn-on voltage is obtained by measuring the gate to source voltage of the device when a predetermined drain current flows through the device under test. Therefore, we can write the turn-on voltage, V_T, as

$$V_T = V_{th} + \sqrt{\frac{2I_{ds}}{\beta_{eff}}} \tag{3.5}$$

$$\beta_{eff} = \mu_{eff}\frac{W}{L}C_{eff}$$

where I_{ds} is the predetermined drain current, μ_{eff} is the effective mobility, W is the width of the device, and L is the length of the device. Combining equation (3.1) and (3.3) yields the equation,

$$V_{th} = \Phi_{gs} - \frac{Q_f}{C_{eff}} - \left[\frac{x_{ob}}{\varepsilon_{ox}} + \frac{x_n - \bar{x}}{\varepsilon_n}\right] Q_N + 2\Phi_F + \frac{\sqrt{4\varepsilon_s q N_B \Phi_F}}{C_{eff}} \tag{3.6}$$

From the equation shown above, the threshold voltage can be changed by changing Q_N, the trapped charges in the nitride layer. During the write operation, electrons will be injected into the nitride layer by tunneling through the tunneling oxide. Since electrons carry negative charges, the increase in electron population in the nitride layer causes the threshold voltage to shift positively. On the other hand, during the erase operation, holes are injected into the nitride layer, causing the threshold voltage to shift negatively.

3.2 Fabrication Sequence

The SONOS transistors are fabricated with the TP300-3 mask sequence developed at Lehigh University's Microelectronics Research Laboratory. The masks are designed to be the device test pattern for various projects. The data represented and the actual devices used in the linear adaptive neuron are taken from the transistor array as shown in figure 3-3. Figure 3-4 shows a close-up view of the entire transistor array. The transistor array is designed to accommodate various gate lengths. The mask lengths designed for the transistor array are 100μm, 50μm, 20μm, 10μm, 7μm, 5μm, 4μm, 3μm, 2μm, and 1μm.

The n-channel transistor fabrication sequence is listed as follows:

- **Starting Material**: p substrate,<100>,3in,2-3Ω/cm
- **Front/Backside Implantation**
  1. Furnace Clean
  2. 160 Å Pad Oxide (Dry, 950°C , 20 min.)
  3. Implant Front Side (Boron, 32KeV, 1.2×10^{13} cm^2)
  4. Implant Back Side (Boron, 32KeV, 2×10^{15} cm^2)
  5. Furnace Clean
  6. Anneal (Dry N_2, 950°C , 30 min.)
  7. Etch Pad Oxide (BHF 10:1)
- **Field Oxidation**
  1. Furnace Clean
  2. 5000Å Field Oxide (Wet, 1100°C , 50 min.)
- **Photolithography** (N+ S/D)
  1. Apply Photoresist (Baker)
  2. Prebake Photoresist (98°C , 30 min.)
  3. UV Exposure and Development
  4. Postbake Photoresist (120°C , 30 min.)
  5. Etch Field Oxidation (BHF 10:1)

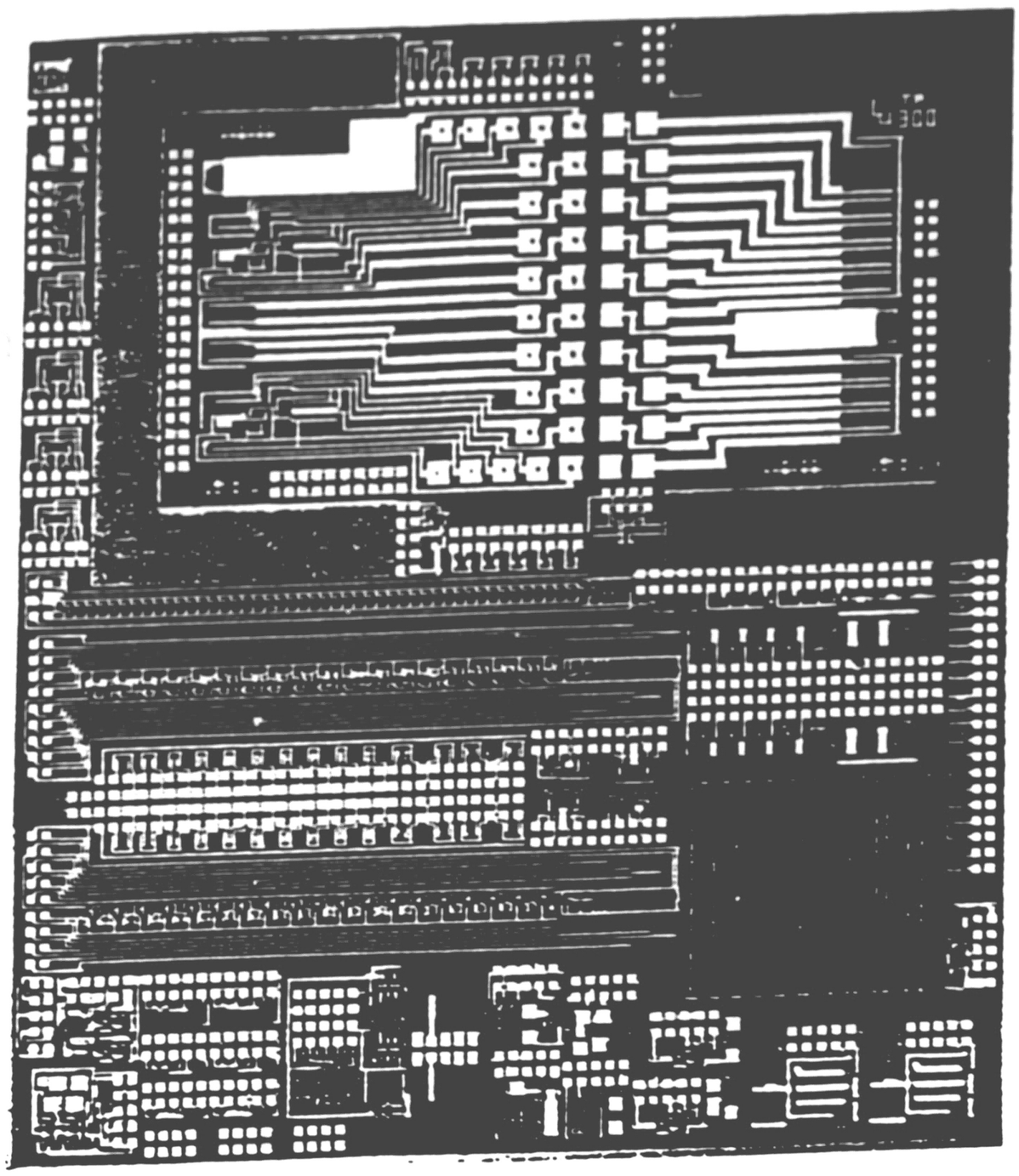


Figure 3-3: Photograph of the TP300 Design

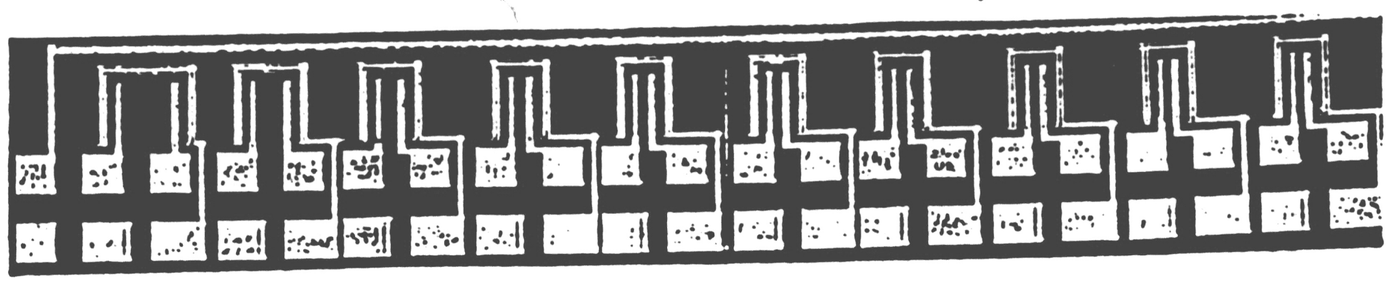


Figure 3-4: Photograph of the transistor array in the TP300

6. Strip Photoresist (PRS-2000)

- **Gate Multi-layer Dielectric**
  1. Furnace Clean
  2. 20 Å Tunneling Oxide (Dry, 720°C , 9 min.)
  3. 120 Å Silicon Nitride (LPCVD, 0.25 torr, 100 sccm NH_3, 20 sccm $SiCl_2H_2$, 735°C , 5 min.)
  4. 40 Å Blocking Oxide (Wet, 1000 °C , 50 min.)
  5. 5000 Å Polysilicon (LPCVD, 0.8 torr, 180 sccm SiH_4, 625°C , 30 min.)
- **Photolithography** (Polysilicon)
  1. Apply Photoresist (Baker)
  2. Prebake Photoresist (98°C , 30 min.)
  3. UV Exposure and Development
  4. Postbake Photoresist (120°C , 30 min.)
  5. Plasma Etch Polysilicon and Triple Dielectric
  6. Strip Photoresist (PRS-2000)
- **Source/Drain Diffusion**
  1. Furnace Clean
  2. Diffusion ($POCl_3$, 900°C , 20 min.)
  3. Drive-in (Dry N_2, 900°C , 30 min.)
  4. Etch p-glass (BHF, 15 sec.)
  5. Furnace Clean
  6. 1200 Å Oxide (Wet, 900°C , 30 min.)
- **Photolithography** (Contact Window)
  1. Apply Photoresist (Baker)
  2. Prebake Photoresist (98°C , 30 min.)
  3. UV Exposure and Development
  4. Postbake Photoresist (120°C , 30 min.)
  5. Etch Oxide (BHF 10:1)
  6. Strip Photoresist (PRS-2000)
- **Metallization** RF Magnetrum Sputtered Aluminum
- **Photolithography** (Metal)

1. **Apply Photoresist (Baker)**
2. Prebake Photoresist (98°C , 30 min.)
3. UV Exposure and Development
4. Postbake Photoresist (120°C , 30 min.)
5. PAN Etch
6. Strip Photoresist (PRS-2000)

- **Backside Metallization**
  1. Plasma Etch Backside
  2. RF Magnetrum Sputtered Aluminum
- **Post Metallization Anneal** PMA (H_2/N_2, 400°C , 30 min.)

The fabrication sequence listed above produces a n-channel transistor only. In order to to produce a p-channel transistor, we can start the fabrication sequence with a n substrste material or use an n well full CMOS fabrication sequence. The CMOS sequence requires more masking steps and implantations and thus requires more time to finish a full run.

3.3 Characterization of the SONOS Devices

To fully characterize the electrical performance of the SONOS devices, several measurements must be carried out. The data presented in this thesis are taken from the SONOS nonvolatile memory transistor with the optical designed length of 100 μm, the optical designed width of 150 μm, and the gate dielectric dimensions of 20 Å of tunneling oxide, 95 Å of silicon nitride, and 25 Å of blocking oxide. Typical measurement techniques will be discussed at the following sections.

3.3.1 High Frequency C-V Measurements

The high frequency C-V measurement can be used to determine the effective capacitance of the device under test. During measurement, a stepping voltage source with a high frequency small AC signal (1 MHz) riding on top of the stepping voltage source is applied to either the gate terminal or the bulk terminal and the measured differential capacitance of the device is recorded. In our measurement, the high frequency C-V curve is obtained by using HP4280A 1 MHz C-V meter with HPIB interface to a HP9836 Computer. Figure 3-5 shows the block diagram of the high frequency C-V measurement setup. A typical high frequency C-V curve is shown in figure 3-6.

We can expressed the capacitance of the device as

$$C_{mea} = \frac{C_{eff} \cdot C_D}{C_{eff} + C_D} \tag{3.7}$$

where C_{eff} is the effective capacitance defined in equation (3.2) and C_D is the depletion capacitance given by

$$C_D = \frac{\varepsilon_s}{x_d} \tag{3.8}$$

where x_d is the depletion width of the device. The device under test is first biased into inversion, then the stepping voltage moves the device toward depletion and accumulation. In accumulation, the capacitance measured corresponds to the effective capacitance of the device because there is no depletion width present in the device. When the device moves toward depletion, the depletion width of the device starts to increase, thereby corresponds to a

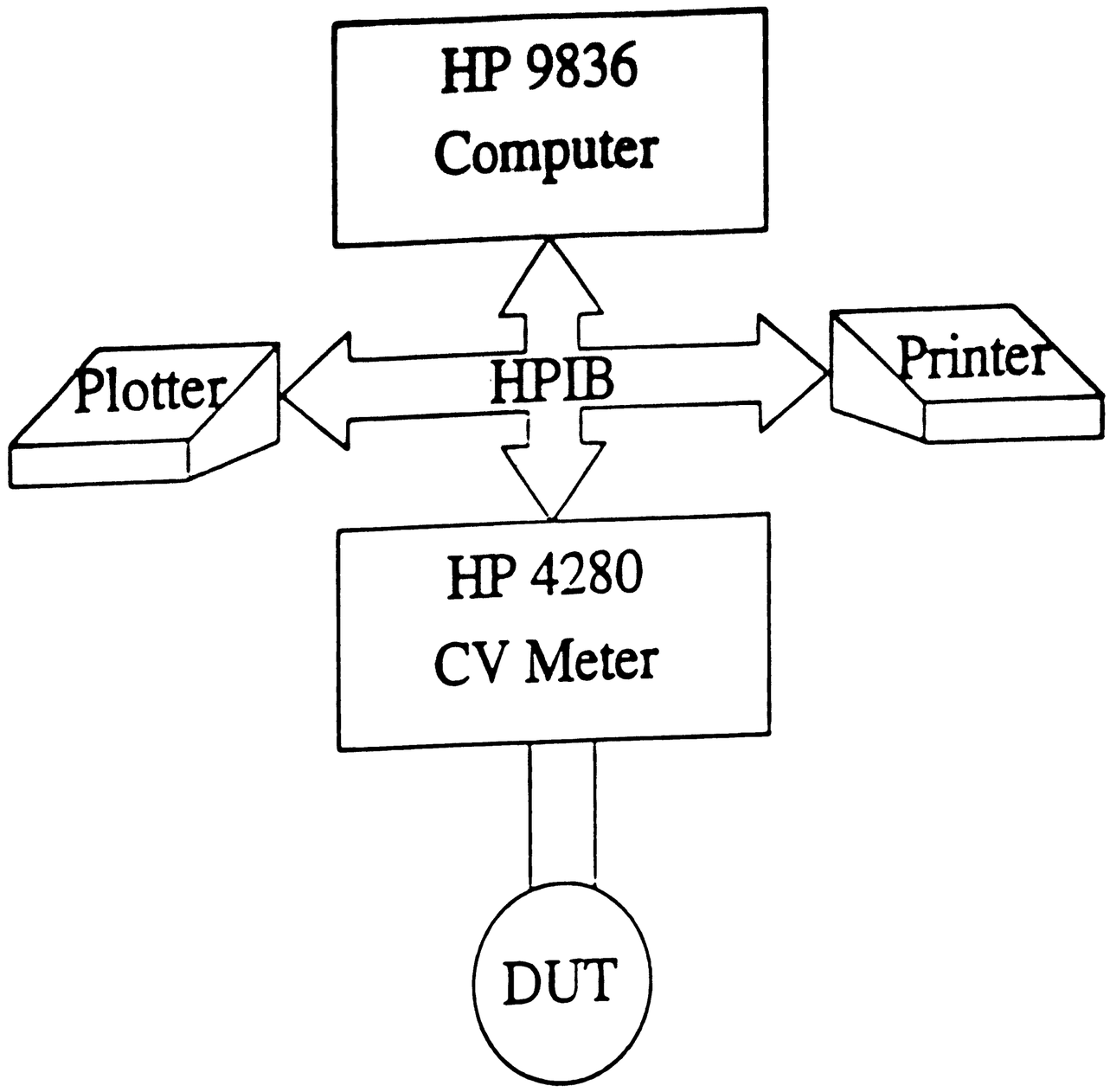


Figure 3-5: Block Diagram of the High Frequency C-V Measurement Setup

decrease in the measured capacitance. Since the small high frequency signal is fast compared to the response time of the minority carriers, the device under test is not able to invert the surface of the device. Therefore, we have a steady increase of the depletion width of the device until the steady state depletion width is reached, which corresponds to C_{min}' in figure 3-6. The effective capacitance result can be used to check against the results from film thickness measurements during fabrication, such as ellipsometry measurements.

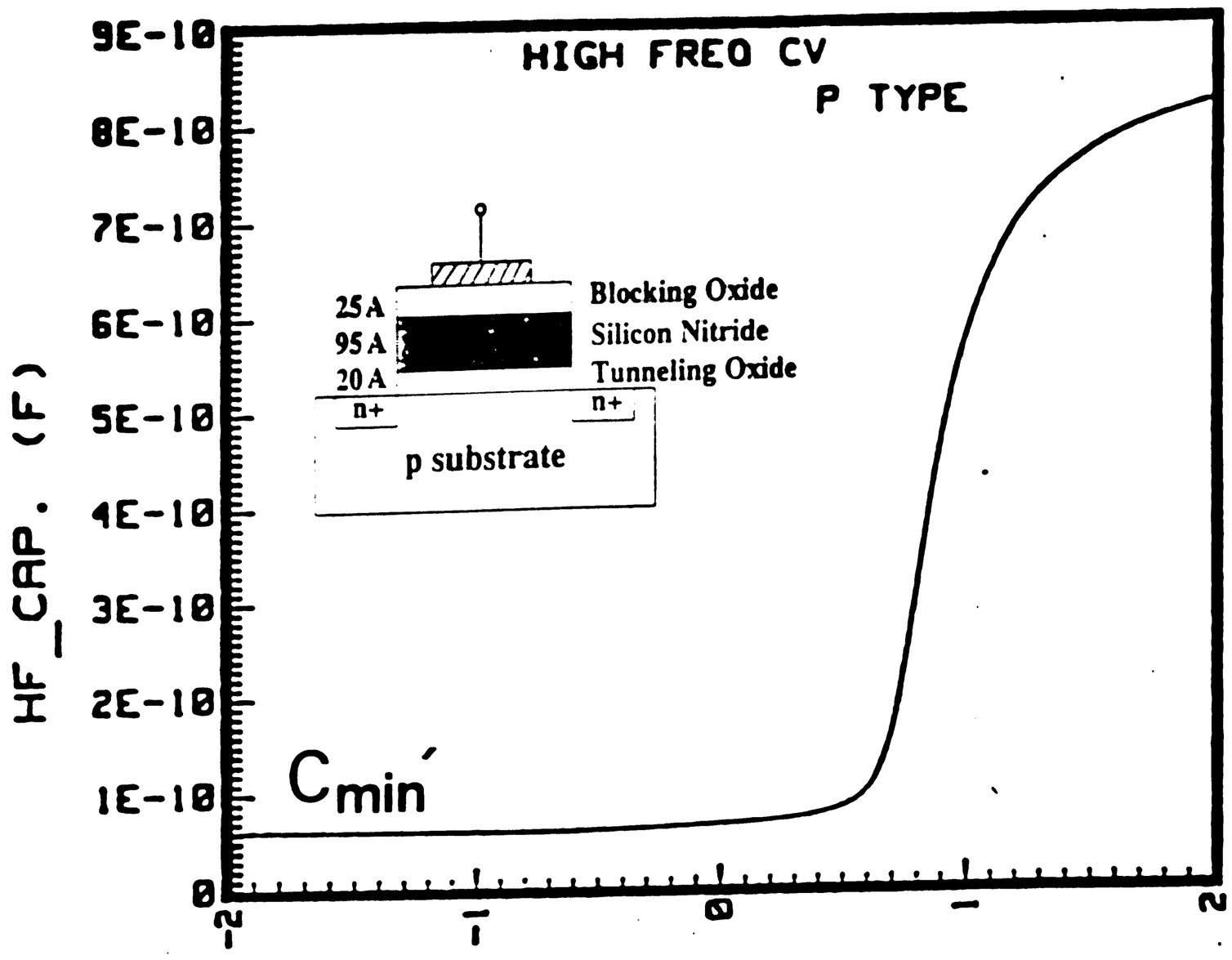


Figure 3-6: High Frequency C-V Curve of a SONOS device[17]

3.3.2 Linear Voltage Ramp Technique

A Linear Voltage Ramp (LVR) technique is used as a quasistatic C-V measurement to determine the memory window of the SONOS device. Unlike the high frequency C-V measurements described in the previous section, the frequency of the ramp is slow enough so that the minority carriers can respond to the change of the bias voltage. In our measurements, the test are performed with the LVR setup which consists of HP9836 computer, HP 8116A Function Generator, A/D and D/A converters, etc. as shown in figure 3-7.

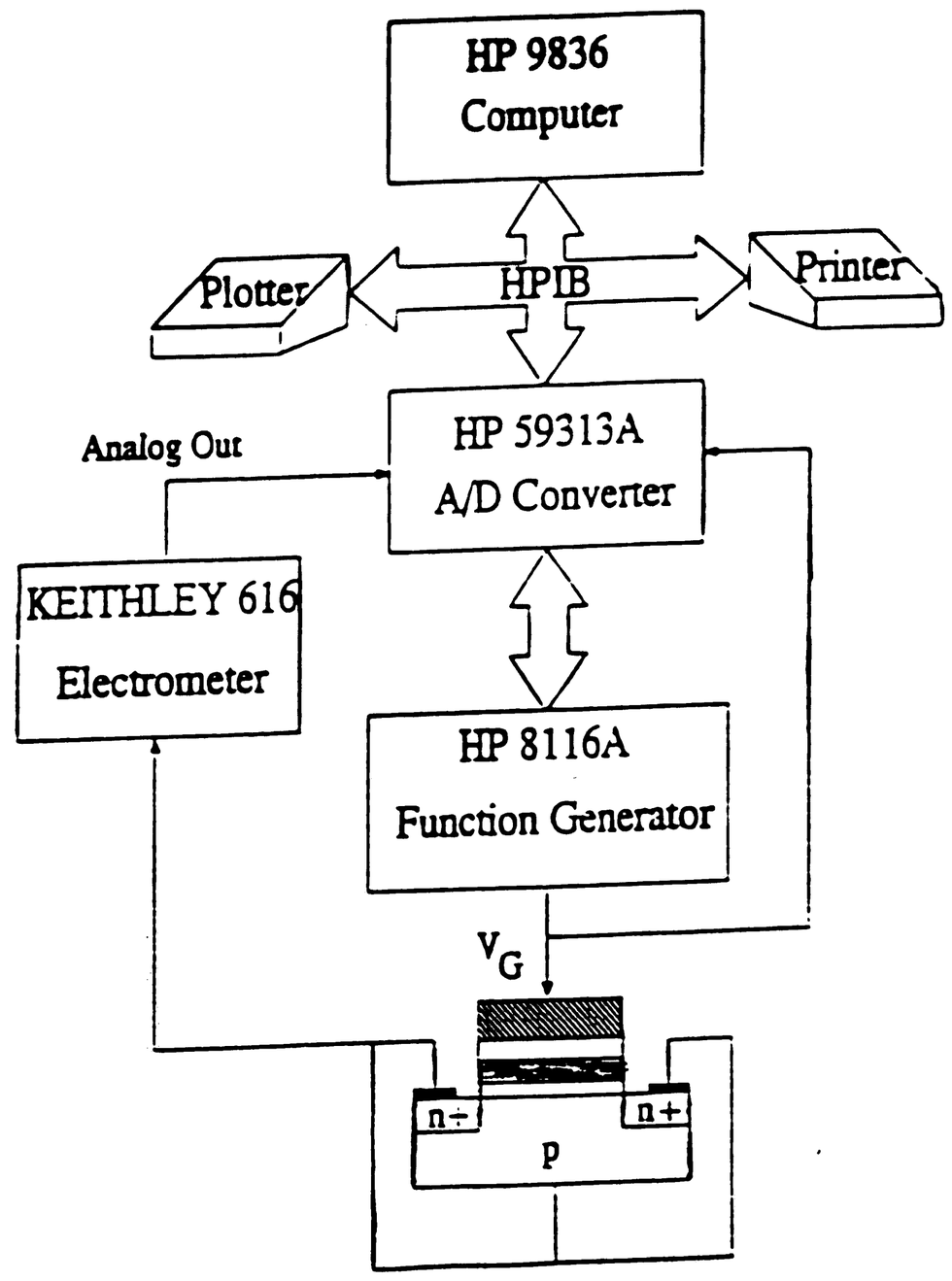


Figure 3-7: Block Diagram of the Linear Voltage Ramp Measurement Setup

The LVR technique utilizes a ramping voltage source applied either to the gate or the bulk terminals and measures the displacement current from the gate or the bulk terminal. The drain and the source terminals are normally tied

together to the bulk terminal and, thus, serve as the source of supplying minority carriers. The voltage ramp which biases the device can be written as

$$V_{GB} = \alpha \cdot t + V_0 \tag{3.9}$$

where α is the ramp rate in units of mV/sec, and t is the time. Notice, that α is positive if the voltage bias is ramping up and α is negative when the voltage bias is ramping down. A DC bias, V_0 may be present, however, it is normally set to zero. The gate or bulk displacement current is monitored during the measurement and is recorded along with the bias voltage value. The gate displacement current can be expressed as

$$\begin{aligned} I_G &= \frac{\partial Q_G}{\partial t} \\ &= \frac{\partial Q_G}{\partial V_{GB}} \cdot \frac{\partial V_{GB}}{\partial t} \\ &= C_{eff} \cdot \alpha \end{aligned} \tag{3.10}$$

where we define the measured capacitance as

$$C_{eff} = \frac{\partial Q_G}{\partial V_{GB}}$$

From equation (3.10), we have a direct relationship between the displacement current and the measured capacitance of the device. Therefore, if we plot the displacement current vs. the bias voltage, we are essentially plotting the effective capacitance vs. the bias voltage.

Figure 3-8 shows a measured curve by the LVR setup. The measurement

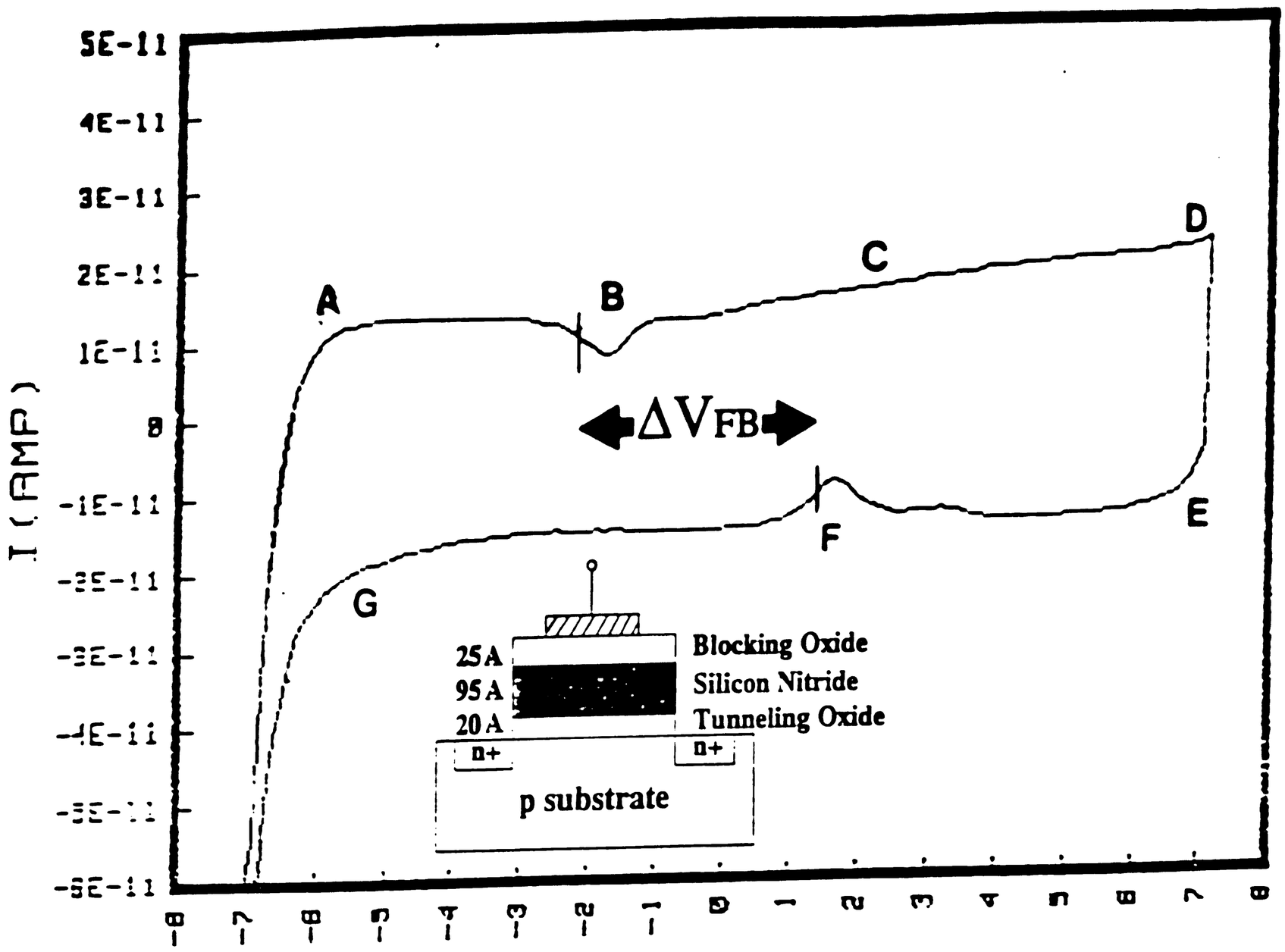


Figure 3-8: Linear Voltage Ramp C-V Curve of a SONOS Device

starts from point A, where the the device is accumulation. When the curve moves from point A to point B, the device is swept from accumulation toward depletion. Since the depletion width increases, the measured capacitance decreases as expected. However, when the device is swept from point B to point C, the device starts to move into inversion. Since the source/drain regions provide minority carriers and the ramping rate is normally slow enough for the

minority carriers to respond, the surface is inverted. Therefore, the measured capacitance starts to increase due to the collapse of the depletion region. The increase in effective capacitance observed from point C to D is caused by the movement of the charge centroid into the nitride. The displacement current reverse its sign when the ramp rate reverses its sign and, thus, the value at point E is just about the opposite of the value at point D. This time, the device starts from inversion and moves towards the accumulation and completes the entire cycle by returning to point A.

The flatband voltage corresponding to the upper trace is found to be different from the flatband voltage in the bottom trace. In an ideal MOS system, these two voltages should lie vertically over one another, however, in the case of a SONOS measurement, the device is written (programmed) during the time the device is under inversion because of the positive gate bias on the device. Therefore, the threshold voltage of the device shifts positively and the effect is clearly shown in the bottom trace. On the other hand, the device is erased (programmed) during the time the device is under accumulation and, thus, the threshold voltage shifts negatively which is self evident in the upper trace. The voltage difference between the two flatband voltages (ΔV_{FB}) is defined as the memory window. The memory window is a measure of the electrical performance of the SONOS device, especially if the SONOS devices are used as digital nonvolatile memory cells.

3.3.3 Dynamic Range Characterization

For an adaptive system to perform effectively, the modifiable element, the nonvolatile analog conductance, should have a wide dynamic range. The dynamic range of the conductance determines how well the circuit adapts under different training conditions. The dynamic range is measured as the ratio of the

highest conductance to the lowest conductance of the device. If we use a digital memory cell as the synaptic element, then 1 bit in the memory cell weight corresponds to 6 dB in dynamic range. Therefore, if we want the weight to have 60 dB in dynamic range, then at least 10 bit of digital memory is required per weight. This drawback in area consumption of the digital weight element has forced the researchers to seek a good analog conductance which has the same dynamic range performance but with much less area.

To measure the dynamic range of the SONOS nonvolatile transistor, we have to measure its channel conductance at the extreme states, namely the fully erased state and the fully written state. The data is taken with a HP4145 Semiconductor Parameter Analyzer controlled by a HP 9836 technical computer running a TECAP (Transistor Electrical Characteristics Analysis Program) software package which is used in data acquisition, parameter extraction and graphics display. The SONOS device is first subjected to a positive gate bias, say 5V, for 5 minutes to ensure the device is saturated at the fully written state. The gate electrode is biased at a low reading voltage, say 1.25V, with the source and bulk terminals grounded. A drain current versus drain to source voltage characteristics is taken with the drain bias swept from -50 mV to 50 mV. The SONOS device is subjected to a negative gate bias (-5V) for 10 minutes to achieve the fully erased state. Again, the drain current versus drain to source voltage characteristics is taken and plotted with the common drain to source voltage axis. Figure 3-9 shows a typical dynamic range characteristics of the SONOS device. Notice, that the SONOS device promises a 60 dB dynamic range between two states.

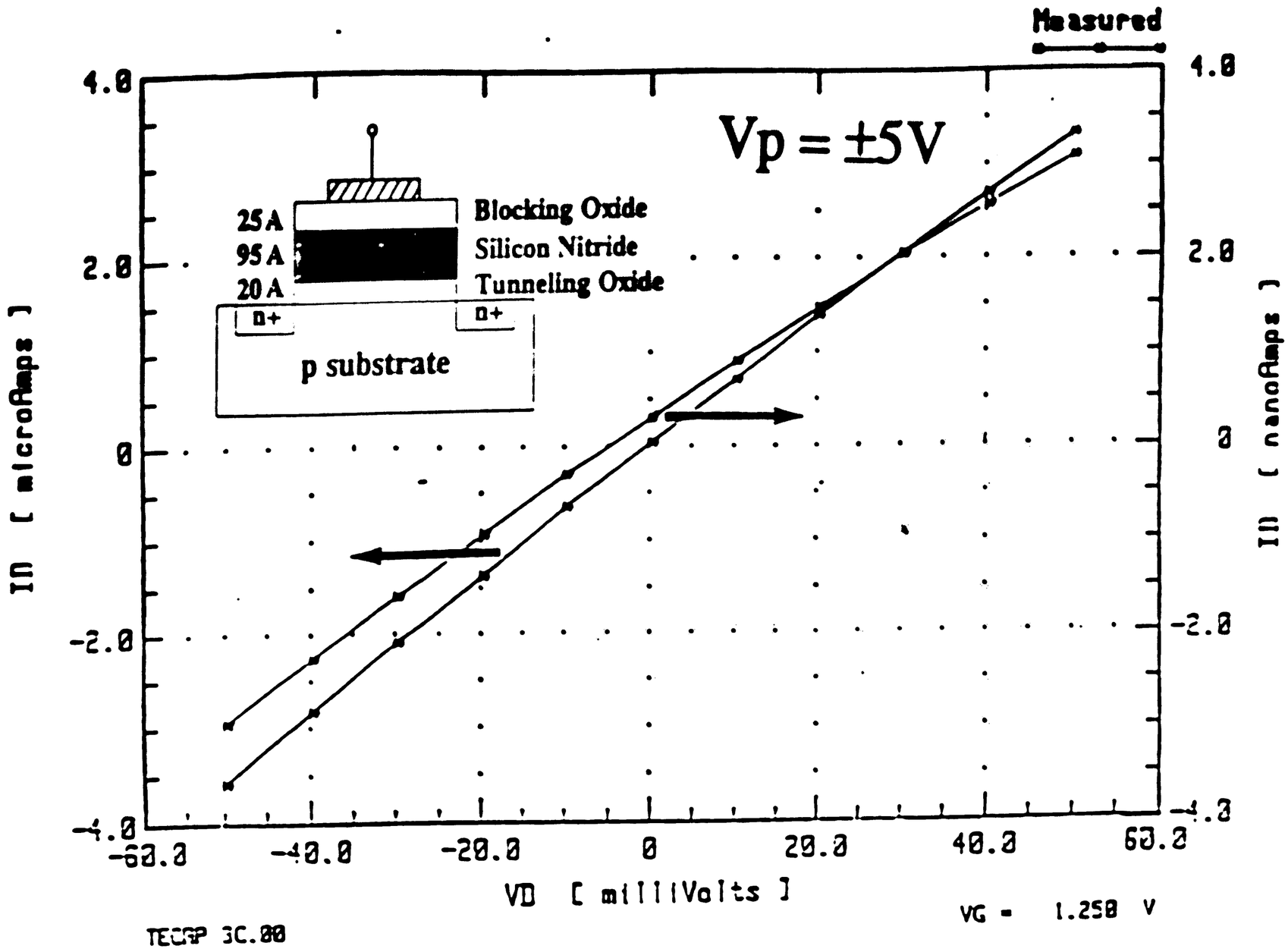


Figure 3-9: Dynamic Range Characteristics of the SONOS Devices W=150 microns, L=100 microns

3.3.4 Erase/Write Characterization

Erase/Write measurements of the SONOS devices provides information on how fast the programmable synaptic weights can be programmed. The erase/write measurement setup is shown in figure 3-10. A resident program is stored in an AIM 65 computer which interfaces with the pattern generator to generate the control waveforms for the switching circuitry in the measurements [18]. Instead of measuring the capacitance of the device, we use three terminals of the device, namely the source, the drain and the gate to measure the turn-on

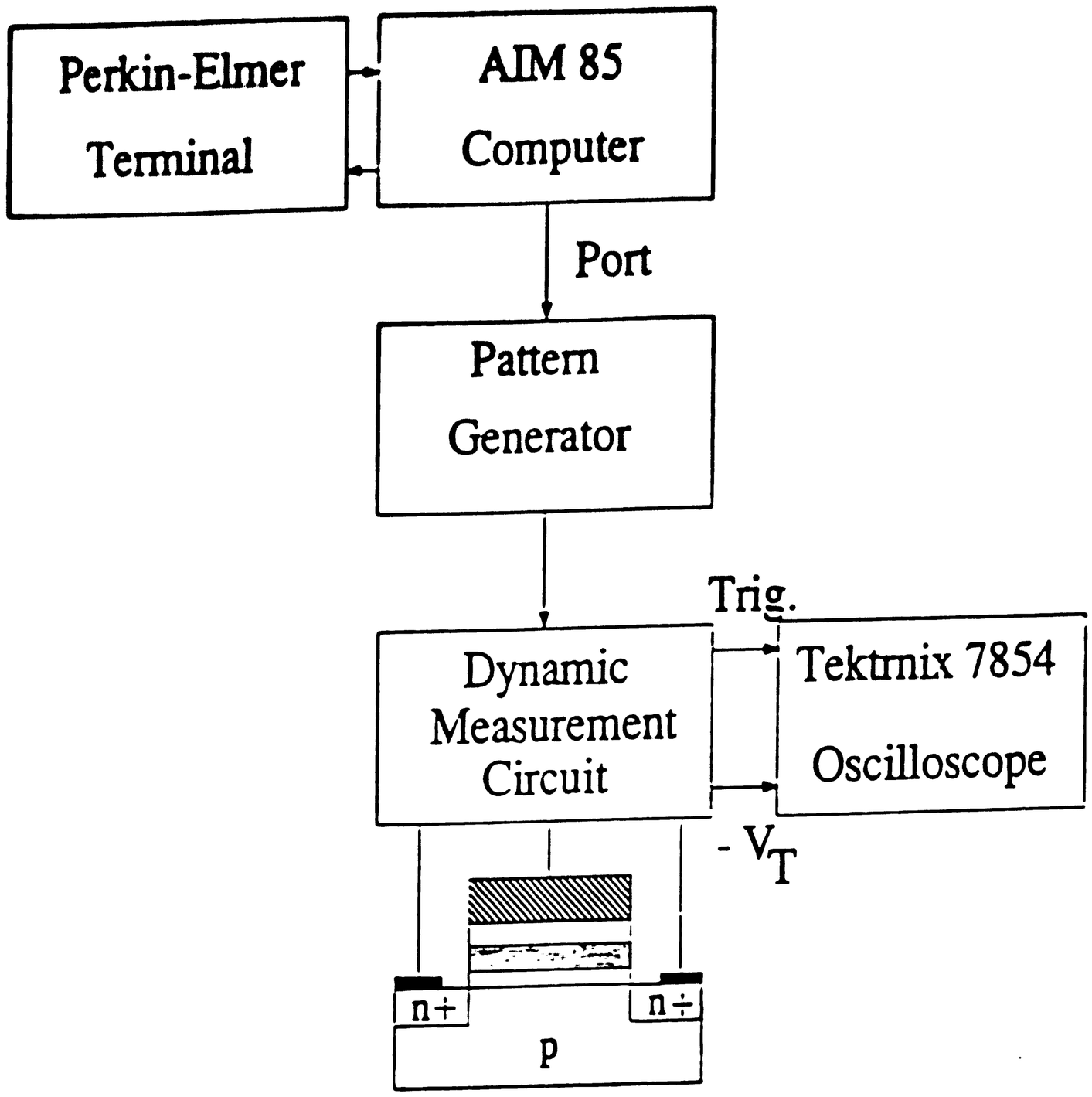


Figure 3-10: Block Diagram of the Dynamic Measurement Setup

voltage of the device. The turn-on voltage of the device has been defined previously in equation (3.5). In this measurement, the drain to source current is set to be 10 μA.

The erase/write characteristics are constructed by plotting the turn-on voltage (thereby, the threshold voltage) vs. the varying pulse width. Normally, the programming voltage is kept constant during these measurements. Figure 3-11 shows the four different modes that a device under test may be subjected to during the measurements. To obtain a write curve, for example, the device is first fully erased by applying a negative gate bias for a long period of time, say 10 seconds. The device is then written with a positive gate bias with a given

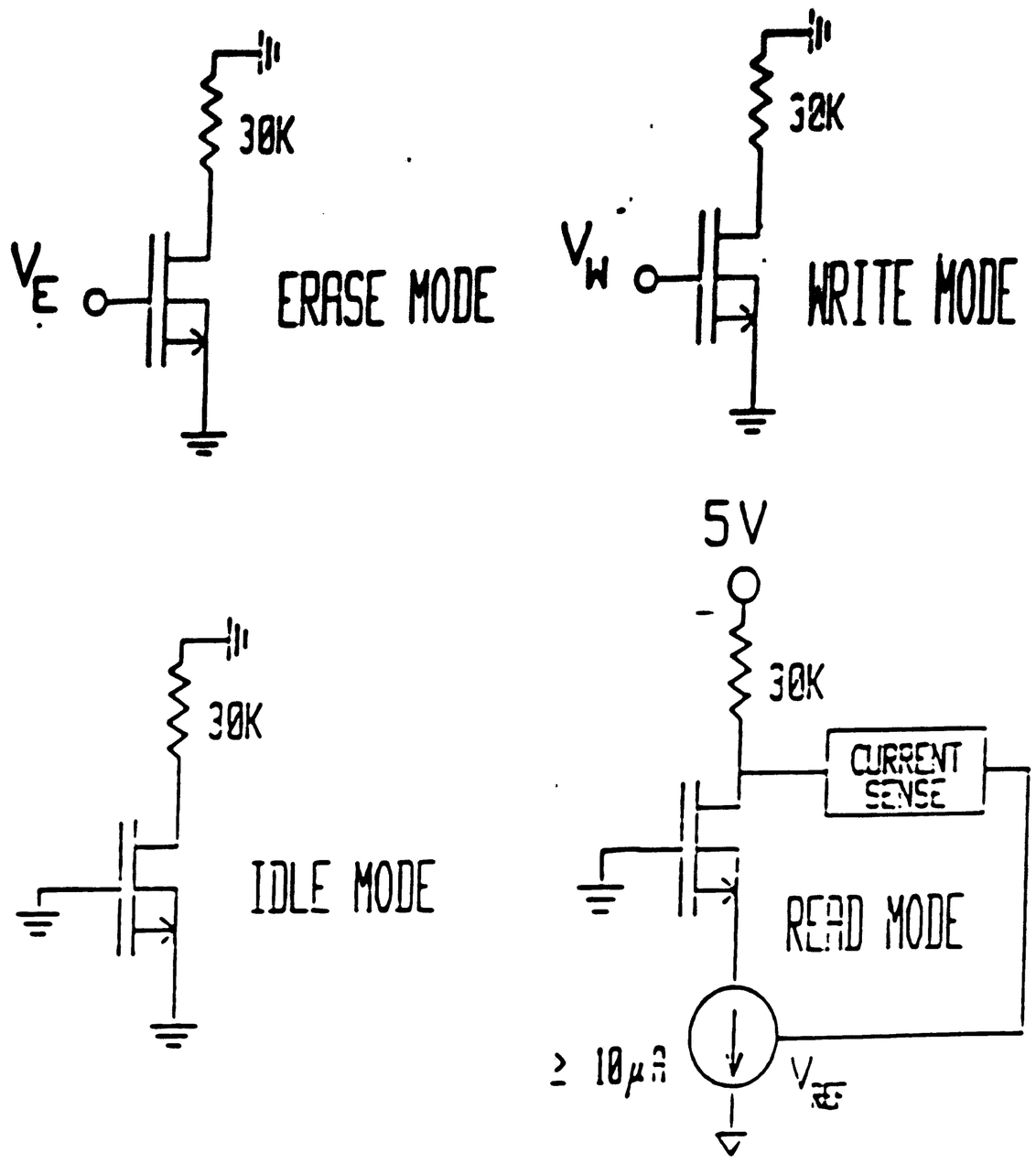


Figure 3-11: Four Testing Modes Performed by the Dynamic Measurement Setup[18]

pulse width and the turn-on voltage is read out during the read operation. The entire procedure repeats as we gather the turn-on voltage data points for various write pulse widths.

Figure 3-12 shows a typical erase/write curve of a SONOS device. As we may expect it, the turn-on (and thereby, threshold) voltage shift more positively as the write pulse width increases. However, for small write pulse widths, the SONOS device does not really response to the programming pulses. On the other hand, if the write pulses are too long, then the device is driven into fully written state. The erase curve is just the opposite case of the write curve. The pulse width that corresponds to the point where the erase curve and the write curve cross each other is called the cross-over time, T_c. A device with a smaller cross-over time indicates that it can be programmed faster than a device having

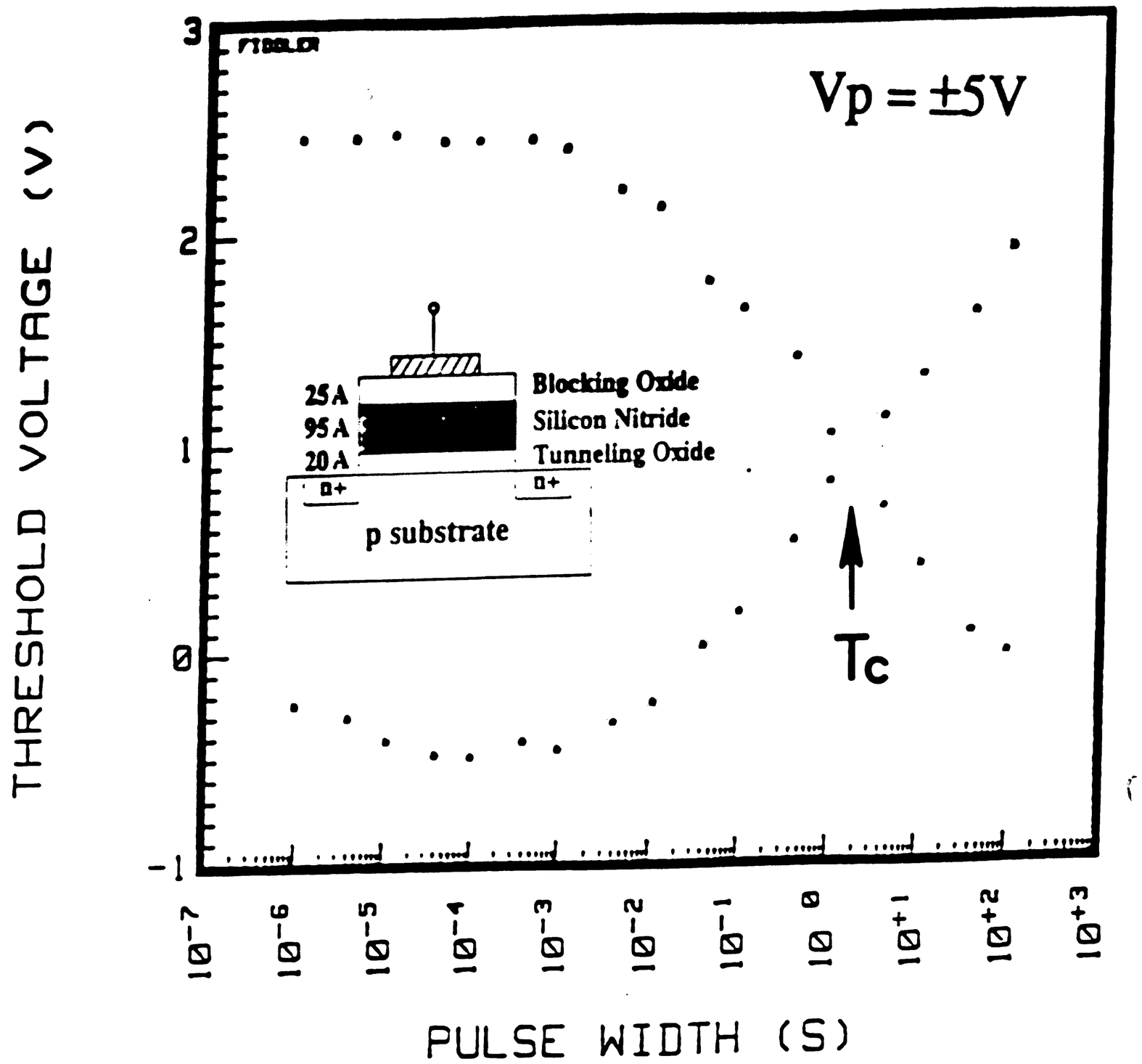


Figure 3-12: Erase/Write Curve of a SONOS Device
W=150 microns, L=100 microns

a larger cross-over time. Higher programming voltages will yield smaller cross-over times due to a larger injection of carriers into the silicon nitride layer. From figure 3-12, the cross-over time of the SONOS devices is in the range of seconds with ± 5V programming voltages. Therefore, the SONOS device may not be very responsive when the device is subjected to a programming pulse of one tenth of a millisecond. This insensitivity of the device to short programming pulses will cost the system to converge in a much slower rate than the system with smaller cross-over time devices.

3.3.5 Retention Characterization

Retention measurements on the SONOS devices provide information on how well the SONOS devices retain their trapped charges in the nitride area. In this measurement, the same setup for the erase/write measurement can be used again for retention measurement with a slight modification to the control sequences sent to the pattern generator. The retention characteristics can be obtained by plotting the turn-on voltage vs. various delay time. The write curve is constructed by first fully writing the device. The threshold voltage is then monitored with various delay times to determine how the threshold voltage decays with respect to time delay. The erase curve is again an opposite case to the write curve. A typical retention measurement is shown in figure 3-13. This curve can be used to project the memory retention after a long period of time delay, say 10 years. In this curve, we can preserve 30% of the original memory information after a projected 10 year period.

One trade-off has to be made between good programming speed and good retention. In order to make the SONOS devices program faster, the tunneling oxide thickness must be reduced. However, a thinner tunneling oxide would also increase the probability of the trapped charge back-tunneling into the bulk silicon. Therefore, the retention performance on a thin tunneling oxide SONOS device is degraded. A more detail study and optimization of the SONOS device has to be made to obtain a good balance between the programming speed and memory retention.

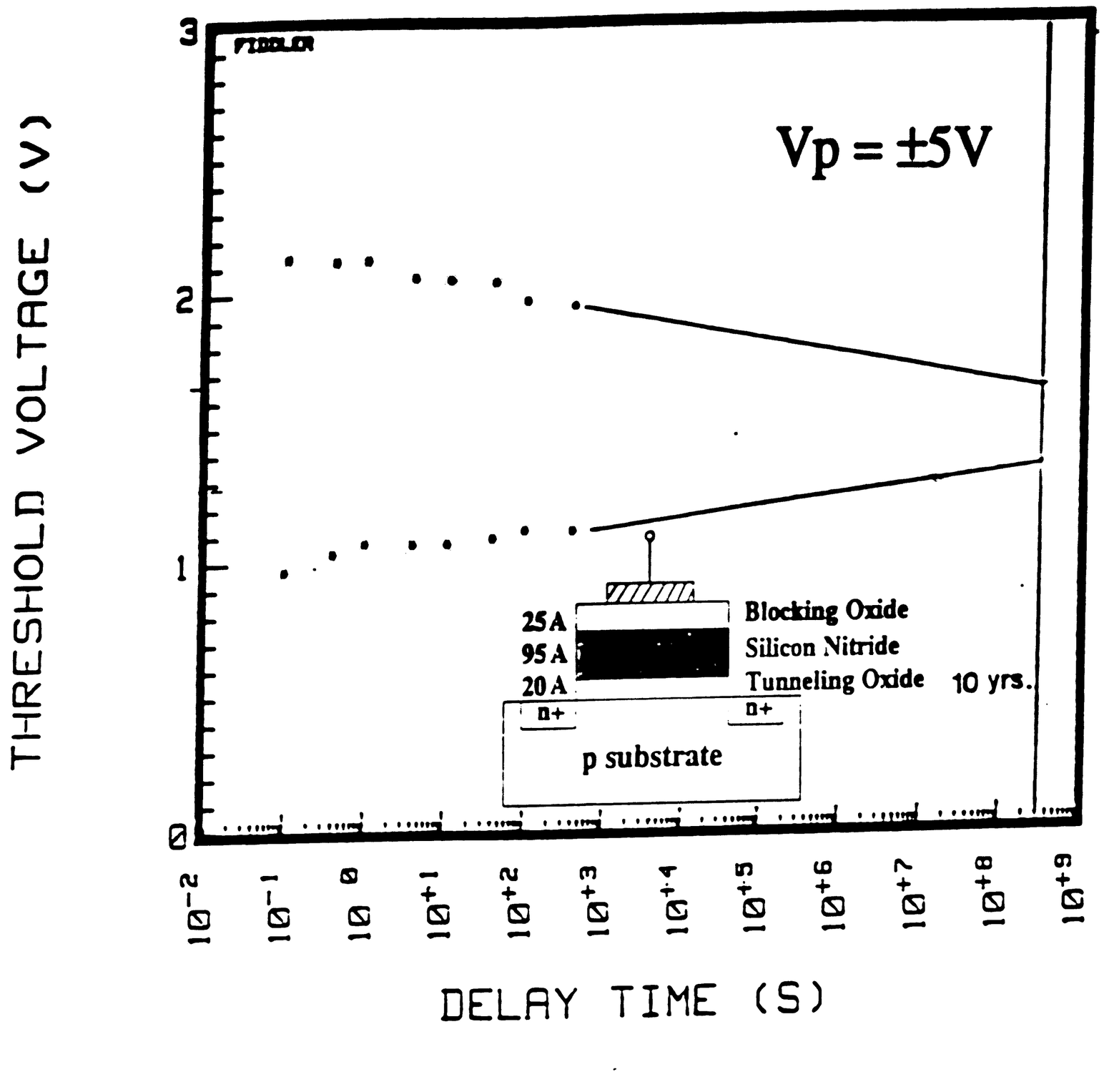


Figure 3-13: Retention Curve W=150 microns, L=100 microns

Chapter 4
Theoretical Analysis of the Linear Adaptive Neuron

4.1 Operational Theory

In this section, a derivation of the performance of a two-tap weight linear adaptive neuron with the LMS and the clipped-data LMS algorithms in the continuous time domain is presented. The LMS algorithm has been proven to work for stationary signal. This derivation is based on sinusoidal signals with and without the presence of Gaussian narrow-band noise. The sinusoidal signals are chosen because we can readily verify the theoretical results against actual experimental results. The derivation involves a training (desired) signal that has a phase and magnitude difference with respect to the input signal. The analog delay line of the input signal is implemented in such a way that the phase difference between the two tapped signals is adjustable by varying the sampling clocks.

Let us begin by writing the mathematical expression for the input and desired signals as

$$d(t) = d_a \cos(\omega_s t + \phi) \tag{4.1}$$

$$x_0(t) = x_a \cos \omega_s t + n_0(t)$$

$$x_1(t) = x_a \cos \omega_s (t - T) + n_1(t)$$

where $x_0(t)$, $x_1(t)$ are the input tapped signals at time t, $d(t)$ is the desired signal at time t, d_a and x_a are the desired and input signal magnitudes, $n_0(t)$ and $n_1(t)$ are the noise associated with x_0 and x_1 at time t, T is the delay time, and ω_s is the

input signal frequency. In the following subsections, derivations for LMS and clipped-data LMS learning algorithm will be given separately.

4.2 Theoretical Analysis of the Two-Tap Weight Linear Adaptive Neuron with the LMS Learning Algorithm

As previously mentioned in Chapter 2, the weight adjustment can be expressed as

$$\frac{dW}{dt} = -\mu \nabla_W E(\varepsilon^2) \tag{4.2}$$
$$= -\mu \nabla_W \overline{\varepsilon^2}$$

with ∇_W expressed as

$$\nabla_W E\left(\varepsilon_m^2\right) = -2P + 2RW \tag{4.3}$$

where P is the cross-correlation vector between the desired response and the input signal and R is the input correlation matrix. Combining equation (4.2) and (4.3) yields the following equation

$$\frac{dW}{dt} = 2\mu P - 2\mu R W \tag{4.4}$$
$$= 2\mu E\begin{pmatrix} d_m x_m \\ d_m x_{m-1} \end{pmatrix} - 2\mu E\begin{pmatrix} x_m x_m & x_m x_{m-1} \\ x_{m-1} x_m & x_{m-1} x_{m-1} \end{pmatrix} W$$

where P and R are substituted with the definition in Chapter 2. The expected value of a quantity is computed by taking the average value of the particular quantity. Therefore, for the P vector, the expected value is computed as

$$E(dx_0) = \frac{1}{T_s}\int_0^{T_s} (d_a \cos(\omega_s t+\phi))(x_a \cos\omega_s t + n_0(t))\,dt \qquad (4.5)$$
$$= \frac{1}{T_s}\left\{ \int_0^{T_s} d_a x_a \cos(\omega_s t+\phi)\cos\omega_s t\,dt + \int_0^{T_s} d_a \cos(\omega_s t+\phi) n_0(t)\,dt \right\}$$
$$= \frac{1}{T_s}\int_0^{T_s} d_a x_a \cos(\omega_s t+\phi)\cos\omega_s t\,dt$$

where T_s is the period of the input signal. Use the trigonometric identity

$$\cos A \cos B = \frac{1}{2}\left[\cos(A-B)+\cos(A+B)\right]$$

Equation (4.5) becomes

$$E(dx_0) = \frac{d_a x_a}{2T_s}\left\{ \int_0^{T_s} \cos\phi\,dt + \int_0^{T_s} \cos(2\omega_s t+\phi)\,dt \right\}$$
$$= \frac{d_a x_a}{2}\cos\phi$$

Other elements in the matrix R and vector P are computed in much the same way to attain the following results

$$P = \begin{pmatrix} \frac{d_a x_a}{2}\cos\phi \\ \frac{d_a x_a}{2}\cos(\phi+\omega_s T) \end{pmatrix} \qquad (4.6)$$

$$R = \begin{pmatrix} \frac{x_a^2}{2}+\sigma_n^2 & \frac{x_a^2\cos(\omega_s T)}{2} \\ \frac{x_a^2\cos(\omega_s T)}{2} & \frac{x_a^2}{2}+\sigma_n^2 \end{pmatrix}$$

where σ_n^2 is the variance of the assumed narrowband Gaussian noise. The

time-varying noise components $n_0(t)$ and $n_1(t)$ are assumed to be uncorrelated and, therefore, we have

$$E\left(n_0^2(t)\right) = E\left(n_1^2(t)\right) = \sigma_n^2 \tag{4.7}$$

$$E\left(n_0(t)\, n_1(t)\right) = 0$$

Combining equations (4.4) and (4.6) yields:

$$\frac{d}{dt}\begin{pmatrix} W_0 \\ W_1 \end{pmatrix} = \mu d_a x_a \begin{pmatrix} \cos\phi \\ \cos(\phi+\omega_s T) \end{pmatrix} - 2\mu \begin{pmatrix} \frac{x_a^2}{2}+\sigma_n^2 & \frac{x_a^2}{2}\cos(\omega_s T) \\ \frac{x_a^2}{2}\cos(\omega_s T) & \frac{x_a^2}{2}+\sigma_n^2 \end{pmatrix} \begin{pmatrix} W_0 \\ W_1 \end{pmatrix} \tag{4.8}$$

In order to solve for the steady state solution of the weight value, the R matrix has to be diagonalized. Therefore, let us introduce a transformation as

$$W = \Phi\eta \qquad \eta = \Phi^{-1} W \tag{4.9}$$

where

$$\Phi = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \quad \Phi^{-1} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = \Phi^T$$

With the matrix transformation, we can diagonalize the R matrix as follow

$$\frac{d}{dt}\left(\Phi^{-1} W\right) = 2\mu\Phi^{-1}P - 2\mu\Phi^{-1}R\Phi\eta$$

and obtain the following expression

$$\frac{d}{dt}\begin{pmatrix}\eta_0\\ \eta_1\end{pmatrix} = \frac{\mu d_a x_a}{\sqrt{2}}\begin{pmatrix}\cos\phi-\cos(\phi+\omega_s T)\\ \cos\phi+\cos(\phi+\omega_s T)\end{pmatrix} \qquad (4.10)$$
$$-2\mu\begin{pmatrix}\frac{x_a^2}{2}(1-\cos\omega_s T)+\sigma_n^2 & 0\\ 0 & \frac{x_a^2}{2}(1+\cos\omega_s T)+\sigma_n^2\end{pmatrix}\begin{pmatrix}\eta_0\\ \eta_1\end{pmatrix}$$

The steady state solution of the weight values can be determined by first setting equation (4.10) to zero and thus find the steady state solution for the quantities η_0 and η_1. The steady state solution of the transformed weight values can be then obtained by utilizing equation (4.9).

$$\eta_0 = \frac{d_a x_a[\cos\phi-\cos(\phi+\omega_s T)]}{\sqrt{2}[x_a^2(1-\cos\omega_s T)+2\sigma_n^2]} \qquad (4.11)$$
$$\eta_1 = \frac{d_a x_a[\cos\phi+\cos(\phi+\omega_s T)]}{\sqrt{2}[x_a^2(1+\cos\omega_s T)+2\sigma_n^2]}$$

and the weight values become

$$W_0 = \frac{d_a x_a[x_a^2\cos\phi+2\cos\phi\sigma_n^2-x_a^2\cos\omega_s T\cos(\phi+\omega_s T)]}{[x_a^2+2\sigma_n^2]^2-x_a^4\cos^2\omega_s T} \qquad (4.12)$$
$$W_1 = \frac{d_a x_a[x_a^2\cos(\phi+\omega_s T)+2\cos(\phi+\omega_s T)\sigma_n^2-x_a^2\cos\omega_s T\cos\phi]}{[x_a^2+2\sigma_n^2]^2-x_a^4\cos^2\omega_s T}$$

Equation (4.12) shows the general expression for the steady solution of the weight value. For the special case where $\omega_s T$ is equal to $\pi/2$ (or $(2n+1)\pi/2$), we have a 90 degree phase shift between $x_0(t)$ and $x_1(t)$. Equation (4.12) can be simplified to

$$W_0 = \frac{d_a x_a [x_a^2 \cos\phi + 2\cos\phi\,\sigma_n^2]}{[x_a^2 + 2\sigma_n^2]^2} \tag{4.13}$$

$$W_1 = \frac{-d_a x_a [x_a^2 \sin\phi + 2\sin\phi\,\sigma_n^2]}{[x_a^2 + 2\sigma_n^2]^2}$$

Further simplification can be made if the noise is neglected in formulation of the equation. If we ignore the contribution of the noise in this derivation, the steady state weight values can be expressed as

$$W_0 = \frac{d_a \cos\phi}{x_a} \tag{4.14}$$

$$W_1 = \frac{-d_a \sin\phi}{x_a}$$

The derivation for the analysis for the clipped-data LMS error algorithm is quite similar as described above. The distinct difference between the analysis with the LMS error algorithm and the clipped-data LMS error algorithm is outlined in the next section.

4.3 Theoretical Analysis of the Two-Tap Weight Linear Adaptive Neuron with the Clipped-data LMS Learning Algorithm

The difference between the LMS error algorithm and clipped-data LMS error algorithm is the amplitude information of the input signal is not employed in the computation of the incremental weight update values. Thus, P and R become

$$P = E\begin{pmatrix} d_m\, sgn\,[x_m] \\ d_m\, sgn\,[x_{m-1}] \end{pmatrix} \tag{4.15}$$

$$R = E\begin{pmatrix} x_m\, sgn[x_m] & x_m\, sgn[x_{m-1}] \\ x_{m-1}\, sgn[x_m] & x_{m-1}\, sgn[x_{m-1}] \end{pmatrix}$$

The matrix element of the autocorrelation matrix R can be calculated as

$$E\left(x_0\, sgn[x_0]\right) = \frac{1}{T_s}\int_0^{T_s} (x_a \cos\omega_s t + n_0(t))\, sgn\,(x_a \cos\omega_s t + n_0(t))\, dt \tag{4.16}$$

and with the identity

$$sgn\,(A+B) = \frac{A\, sgn(A) + B\, sgn(B)}{A+B}$$

equation (4.16) can be rewritten as

$$\begin{aligned} E\left(x_0\, sgn(x_0)\right) = \frac{1}{T_s} &\left\{ \int_0^{T_s} (x_a \cos\omega_s t)\, sgn\,(x_a \cos\omega_s t)\, dt \right. \\ &\left. + \int_0^{T_s} (n_0(t))\, sgn\,(n_0(t))\, dt \right\} \\ = \frac{4x_a}{T_s}&\int_0^{T_s} \cos\omega_s t + \frac{2\sqrt{2}}{\pi}\sigma_n \\ = \frac{2x_a}{\pi} &+ \frac{2\sqrt{2}}{\pi}\sigma_n \end{aligned}$$

where

$$E\left(n_0(t)\, sgn[n_0(t)]\right) = \frac{2\sqrt{2}}{\pi}\sigma_n$$

We can proceed with the calculations of the elements for P and R as described in the previous section.

$$P = \frac{2d_a}{\pi}\begin{pmatrix} cos\phi \\ cos(\omega_s T+\phi)\end{pmatrix} \tag{4.17}$$

$$R = \frac{2}{\pi}\begin{pmatrix} x_a+\sqrt{2}\sigma_n & x_a cos\omega_s T \\ x_a cos\omega_s T & x_a+\sqrt{2}\sigma_n \end{pmatrix}$$

Following the procedures outlined in the previous section, the gradient of the weight vector can be written as

$$\frac{d}{dt}\begin{pmatrix} W_0 \\ W_1\end{pmatrix} = \frac{4\mu d_a}{\pi}\begin{pmatrix} cos\phi \\ cos(\omega_s T+\phi)\end{pmatrix} - \frac{4\mu}{\pi}\begin{pmatrix} x_a+\sqrt{2}\sigma_n & x_a cos\omega_s T \\ x_a cos\omega_s T & x_a+\sqrt{2}\sigma_n \end{pmatrix}\begin{pmatrix} W_0 \\ W_1\end{pmatrix} \tag{4.18}$$

and using the matrix transformation to diagonalize the R matrix we arrive at

$$\frac{d}{dt}\begin{pmatrix} \eta_0 \\ \eta_1\end{pmatrix} = \frac{4\mu d_a}{\sqrt{2}\pi}\begin{pmatrix} cos\phi - cos(\phi+\omega_s T) \\ cos\phi + cos(\phi+\omega_s T)\end{pmatrix} - \frac{4\mu}{\pi}\begin{pmatrix} x_a(1-cos\omega_s T)+\sqrt{2}\sigma_n & 0 \\ 0 & x_a(1+cos\omega_s T)+\sqrt{2}\sigma_n \end{pmatrix}\begin{pmatrix} \eta_0 \\ \eta_1\end{pmatrix} \tag{4.19}$$

We are now in the position to solve equation (4.19) for the steady state weight values. Following the procedures of the previous section, the steady state transformed weight vector components become

$$\eta_0 = \frac{d_a[\cos\phi - \cos(\phi+\omega_s T)]}{\sqrt{2}[x_a(1-\cos\omega_s T)+\sqrt{2}\sigma_n]} \qquad (4.20)$$

$$\eta_1 = \frac{d_a[\cos\phi + \cos(\phi+\omega_s T)]}{\sqrt{2}[x_a(1+\cos\omega_s T)+\sqrt{2}\sigma_n]}$$

and the steady state weight vectors are,

$$W_0 = \frac{d_a[x_a\cos\phi+\sqrt{2}\cos\phi\,\sigma_n - x_a\cos\omega_s T\cos(\phi+\omega_s T)]}{[x_a+\sqrt{2}\sigma_n]^2 - x_a{}^2\cos^2\omega_s T} \qquad (4.21)$$

$$W_1 = \frac{d_a[x_a\cos(\phi+\omega_s T)+\sqrt{2}\cos(\phi+\omega_s T)\sigma_n - x_a\cos\omega_s T\cos\phi]}{[x_a+\sqrt{2}\sigma_n]^2 - x_a{}^2\cos^2\omega_s T}$$

For a particular case where the quantity $\omega_s T$ equals $\pi/2$ (or $(2n+1)\pi/2$), equation (4.21) can be simplified to

$$W_0 = \frac{d_a[x_a\cos\phi+\sqrt{2}\cos\phi\,\sigma_n]}{[x_a+\sqrt{2}\sigma_n]^2} \qquad (4.22)$$

$$W_1 = \frac{-d_a[x_a\sin\phi+\sqrt{2}\sin\phi\,\sigma_n]}{[x_a+\sqrt{2}\sigma_n]^2}$$

If we ignore the effect of noise as discussed in the previous section, then the steady state weight solution becomes

$$W_0 = \frac{d_a \cos\phi}{x_a} \qquad (4.23)$$

$$W_1 = \frac{-d_a \sin\phi}{x_a}$$

For the case where no noise is present in the system, the LMS error algorithm has the same steady state weight values as the clipped-data LMS algorithm. For a special configuration where ϕ is equal to 45 degrees, both the steady state weight values are equal to each other.

By solving equation (4.10) and (4.19), the time constant of the transient weight value can be written as

LMS Algorithm

$$\tau_0 = \frac{1}{\mu[x_a^2(1-\cos\omega_s T)+2\sigma_n^2]}$$

$$\tau_1 = \frac{1}{\mu[x_a^2(1+\cos\omega_s T)+2\sigma_n^2]} \qquad (4.24)$$

Clipped-data LMS Algorithm

$$\tau_0 = \frac{1}{\frac{4\mu}{\pi}[x_a(1-\cos\omega_s T)+\sqrt{2}\sigma_n]}$$

$$\tau_1 = \frac{1}{\frac{4\mu}{\pi}[x_a(1+\cos\omega_s T)+\sqrt{2}\sigma_n]}$$

The noise present in the system actually tends to reduce the time constant, which leads to faster convergence of the weight values. However, the steady state weight values steer away from the optimum weight values as shown in equation (4.23) because of the presence of noise.

The convergence factor, μ, controls the speed of the convergence of the algorithm. A large convergence factor will result in a fast adjustment of the weight values to minimize the overall error. However, due to large convergence factor, each weight adjustment becomes so coarse such that every time the weights make an adjustment, the circuit overcorrect itself. This results an oscillation in the error signal and large misadjustment caused by the variance of the weight values. On the other hand, a small convergence factor will minimize the misadjustment of the circuit; however, the fine incremental weight adjustment requires a longer time before the circuit reaches its steady state condition. Therefore, the ideal case would be such that the circuit has a large convergence factor at the initial stage of adaptation, and the convergence factor reduces its value as the error signal reduces. This variable convergence factor scheme would cut down both the convergence time and the misadjustment. In our linear adaptive neuron setup, a variable convergence factor scheme is achieved by combining an unique SONOS synaptic elements property and the signal processing circuitry. In appendix B, a detailed discussion on the variable convergence factor scheme is presented.

Chapter 5
Experimental Setup and Results

5.1 The Linear Adaptive Neuron Description

The linear adaptive neuron circuit is implemented on four different breadboards to provide better isolation between the various parts of the circuit. The circuit is divided into the following sections: the digital control/clocking module, the analog delay line section, the analog signal processor section, and the SONOS synaptic weight section. The common point (ground) for digital subsection and the analog subsection is kept separated except connecting at a single point. Special precaution is taken to provide both clean analog and digital power supplies and to avoid ground loop problems. All the components used in the linear adaptive neuron circuit except the SONOS synaptic weight elements are commercially available. All the digital logics and the analog switches are CMOS components while the operational amplifiers and the comparators are the TTL linear series. The power rails of the circuit are set to be ±7.5V to demonstrate the low programming capability of the SONOS weights and to comply to the voltage handling capability of the CMOS analog switches used in the circuit.

The circuit operation can be divided into three modes, the disabled mode, the initialized mode, and the programmed mode. During the disabled mode, the error feedback path is disconnected from the circuit and the gate electrodes of the SONOS synaptic weight elements are connected to a read voltage in order to read out the channel conductance information of those elements. Since the error feedback path is disconnected, there is no weight update or weight adjustment and, therefore, the learning algorithm is said to be disabled. During the initialized mode, the gate electrodes of the SONOS synaptic weight elements are

connected to an initializing voltage. The polarity of the initializing voltage can be selected by the user through a SPDT switch in order to preset the weight value to a predetermined state. Since the weight value is determined by the differential conductances of two SONOS synaptic weight elements, the predetermined state can be either positive or negative. The programmed mode is the normal mode of operation. It is during the programmed mode where the adaptation takes place. There are two subprocesses under the programmed mode, namely, the updating process and the reading process. During the updating process, a programming voltage is present at the gate electrode of the SONOS synaptic weight and, thus, the threshold voltage of the SONOS synaptic weight element is altered accordingly. During the reading process, a read voltage is presented to the gate electrode instead of programming voltage and, thus, the SONOS synaptic weight element acts as a conductance. Generally, the linear adaptive neuron is first initialized, then it goes through the disable mode, and finally the programmed mode.

The block diagram of the single level linear adaptive neuron circuit is shown in figure 5-1. The input signal, $x(m)$, is first passed through an analog delay line to create two tapped signals, $x_0(m)$ and $x_1(m)$. The two tapped signals are multiplied with their respective weight values, W_0 and W_1, and the results are summed in a summing amplifier. The summed signal is then fed into a correlated double sampling module to remove the unwanted noise and offset voltages of the amplifiers used in the analog delay line and the summing amplifier[19]. The 'clean' signal, called the output signal $y(m)$, is then compared to the training signal, $d(m)$. If there exists any mismatch between the output signal and the training signal, then an error signal is generated as the difference between the output signal and the training signal. The error generated is fed into the clipped data learning algorithm for the calculation of

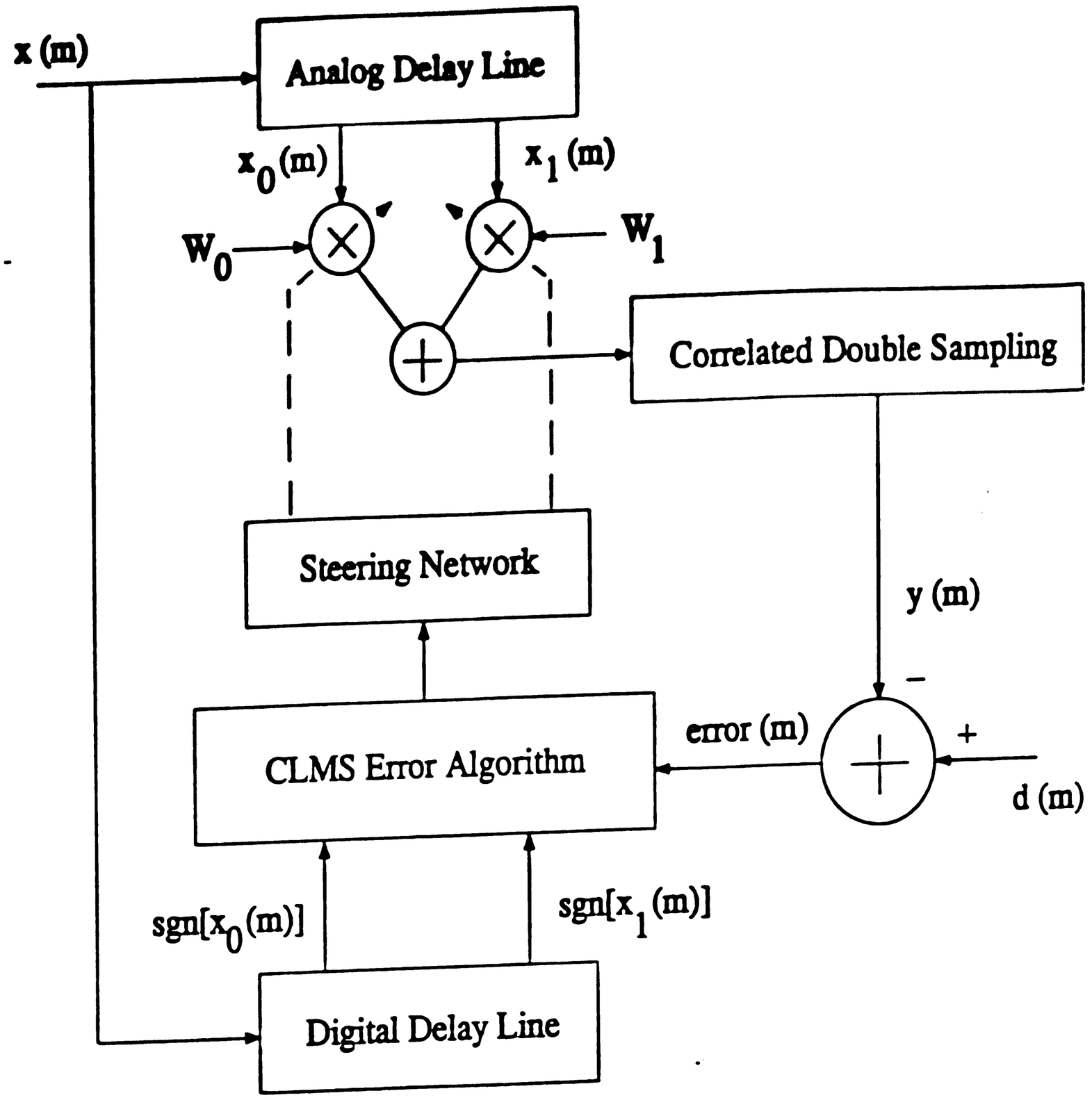


Figure 5-1: Block Diagram of the Single Level Linear Adaptive Neuron

the weight adjustments. A digital delay line is implemented with D flip flops to provide the sign information of the input signal for the learning algorithm. The result of the learning algorithm is fed into the the steering network to steer the proper programming voltage to the appropriate gate electrode of the SONOS synaptic weight elements. In the following sections, various part of the linear adaptive neuron will be discussed.

5.1.1 Digital Control/Clocking Module

The digital control/clocking module generates all the controlling clock signals required to perform different operation modes during adaptation. The clocking scheme controls the position of the analog switches in the circuit and thereby control the signal path within the circuit. Since the circuit is analog in nature with digital clock control signals, the circuit operation may be classified as Discrete Analog Signal Processing (DASP).

Due to the power limitation on the analog switches, the clocking waveform are either unipolar (0 - 7.5V) or bipolar (-7.5 - 7.5V). In the case of a unipolar clock, the low level (logic 0) is represented by 0V and high level (logic 1) is represented by 7.5V. On the other hand, for the case of a bipolar clock, the low level is denoted by -7.5V and high level is denoted by 7.5V. Since the CMOS logic chip used in the digital control module is powered by GND and 7.5V, a level shift on the bipolar waveform is required. The level shifter is done with the comparator which compares the input signal against a reference voltage of 2.5V. When the input signal is in logic 1 (7.5V), the comparator outputs 7.5V and when the input signal is at its logic 0 (0V), the comparator outputs -7.5V. The module is built on a printed circuit board with its own power supply and a decoupling capacitor is provided for each chip used in the module.

The definition of each clock signal is defined in Table 5-1. The timing diagram of the output waveforms is shown in figure 5-2. As shown in Table 5-1, Φ_0, the master clock, is generated from a CMOS timer 555 multivibrator. The oscillation frequency is determined by two external resistors and one external capacitor. All the data presented in this thesis corresponds to a 12.8 KHz master clock frequency. The master clock frequency is divided by 2, 4, and 8 times by a series of T flip-flops to provide the necessary signals for the generation of all other control clocks. If we denote $\Phi_0/2$ as the clock signal twice

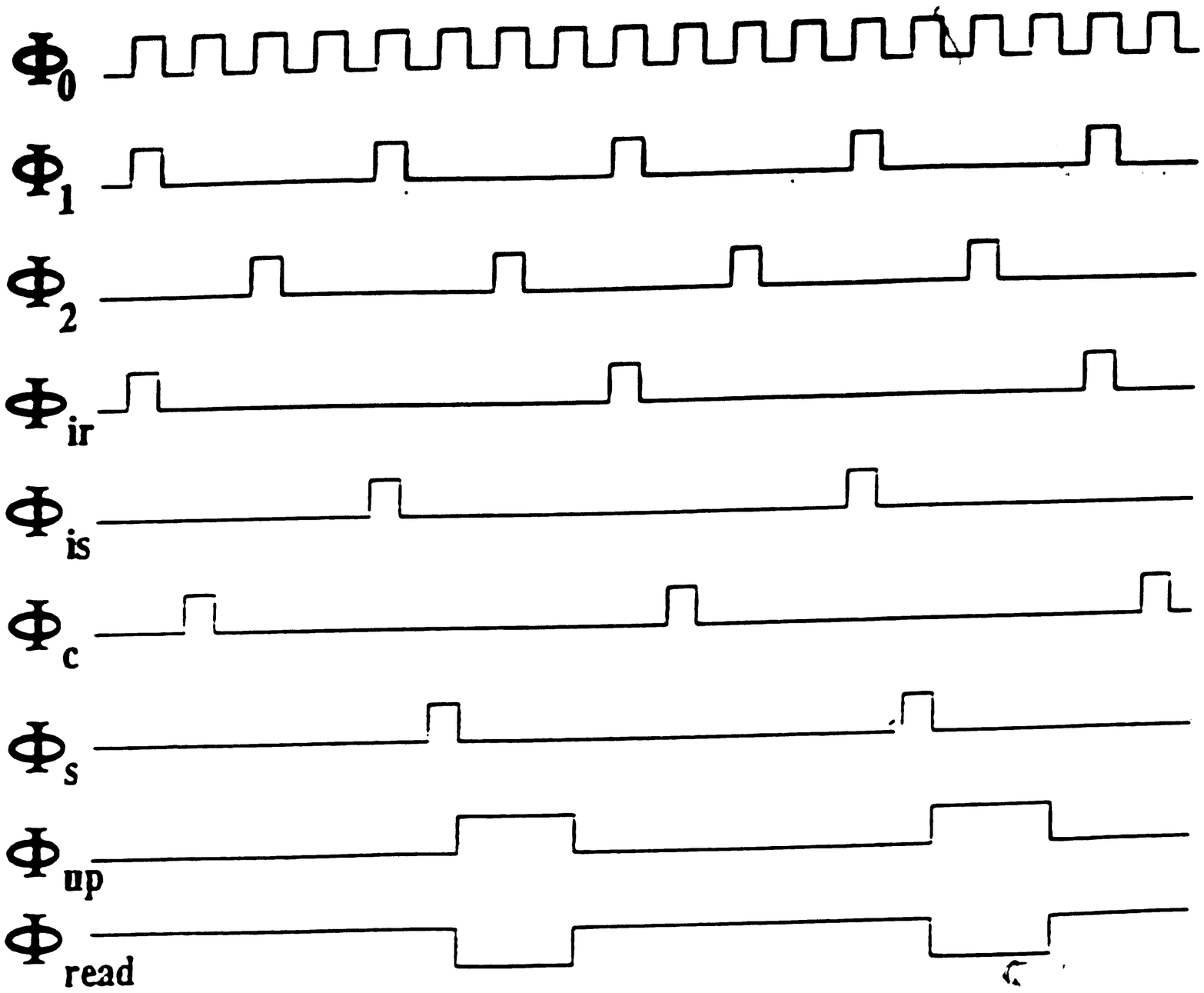


Figure 5-2: Timing Diagram of the Digital Control/Clocking Module

as slow as the master clock, $\Phi_0/4$ as the clock signal four times slower than the master clock and $\Phi_0/8$ as the clock signal eight times slower than the master clock, then we can express all the control clocks by the logical operations listed in Table 5-2 where an overbar denotes the inversion of the clock signal.

The analog delay line sampling clock, Φ_1 and Φ_2, are designed to ensure a two phase, non-overlapping clock for the switched capacitor analog delay line operation. The input clocks, Φ_{ir} and Φ_{is} coincide with Φ_1, but are twice as slow as Φ_1. Therefore, when the Φ_1 is high, either Φ_{ir} or Φ_{is} will be high. When Φ_{ir} is high, the analog delay line receives the reference voltage signal as input. On

Clock Signal Definition		
Singal	Description	Polarity
Φ_0	Master Clock	Unipolar
Φ_1	Sampling Clock for Analog Delay Line	Unipolar
Φ_2	Sampling Clock for Analog Delay Line	Unipolar
Φ_{ir}	Input Reference Clock	Unipolar
Φ_{is}	Input Signal Clock	Unipolar
Φ_c	Clamping Clock (Correlated Double Sampling)	Bipolar
Φ_s	Sampling Clock (Correlated Double Sampling)	Bipolar
Φ_{up}	Weight Updating Clock	Bipolar
Φ_{read}	Weight Reading Clock	Bipolar

Table 5-1: Clock Signal Definition

Clock Signal Generation Operation	
Singal	Logical Operation
Φ_1	$(\Phi_0) \cdot \overline{\Phi}_0 / 2 \cdot \overline{\Phi}_0 / 4$
Φ_2	$(\Phi_0) \cdot \overline{\Phi}_0 / 2 \cdot \Phi_0 / 4$
Φ_{ir}	$(\Phi_1) \cdot \overline{\Phi}_0 / 8$
Φ_{is}	$(\Phi_1) \cdot \Phi_0 / 8$
Φ_c	$(\Phi_0) \cdot \Phi_0 / 2 \cdot \overline{\Phi}_0 / 4 \cdot \overline{\Phi}_0 / 8$
Φ_s	$(\Phi_0) \cdot \Phi_0 / 2 \cdot \overline{\Phi}_0 / 4 \cdot \Phi_0 / 8$
Φ_{up}	$(\Phi_0 / 4) \cdot \Phi_0 / 8$
Φ_{read}	$\overline{\Phi}_{up}$

Table 5-2: Clock Signal Generation Operation

the other hand, when Φ_{is} is high, the analog delay line is fed with the voltage level that is the summation of the reference voltage and the input signal. This

input signal formation scheme is chosen so that either the reference voltage or the summation of the reference voltage and the input signal appears at the tapped location alternatively and therefore, the adaptive signal processor can remove the nonlinear effects caused by excessive drain bias. Φ_c and Φ_s are two clock signals used in the correlated double sampling circuit in the analog signal processor, which will be described in a later section, to remove the nonlinear effects mentioned above. After Φ_s is closed, the output signal is valid and thus the error signal and the weight adjustment can be calculated. Once the weight adjustment is calculated, Φ_{up} will be high to allow the threshold voltage of the SONOS synaptic elements to be altered. While the SONOS synaptic weights are not in the update mode, the Φ_{read} clock is high to read out the conductance information of the SONOS weight elements.

The schematic of the digital control/clocking scheme is shown in figure 5-3. During the read operation of the SONOS synaptic weight elements, a small amount of charge is injected to the nitride layer of the gate dielectric. Although this allows for *read enhancement* of the SONOS weight, excessive read voltages will cause the weight value to drift away from its steady-state optinum value as calculated by the circuit. Therefore, it is necessary to incorporate an idling process for the SONOS devices where all the terminals of the SONOS device are sitting at the same potential. Thus, the programmed mode will consist of three subprocesses: the updating process, the reading process, and the idling process. The incorporation of the idling process requires modification on the switching network described in a later section. A new timing scheme with the idle clock signal and the schematic with EPROM cell are shown in figure 5-4 and 5-5. All the data presented in this thesis were taken without the idle clock signal.

In order to sample the input waveform sufficiently, the Nyquist theorem requires that the sampling frequency be at least twice the input signal

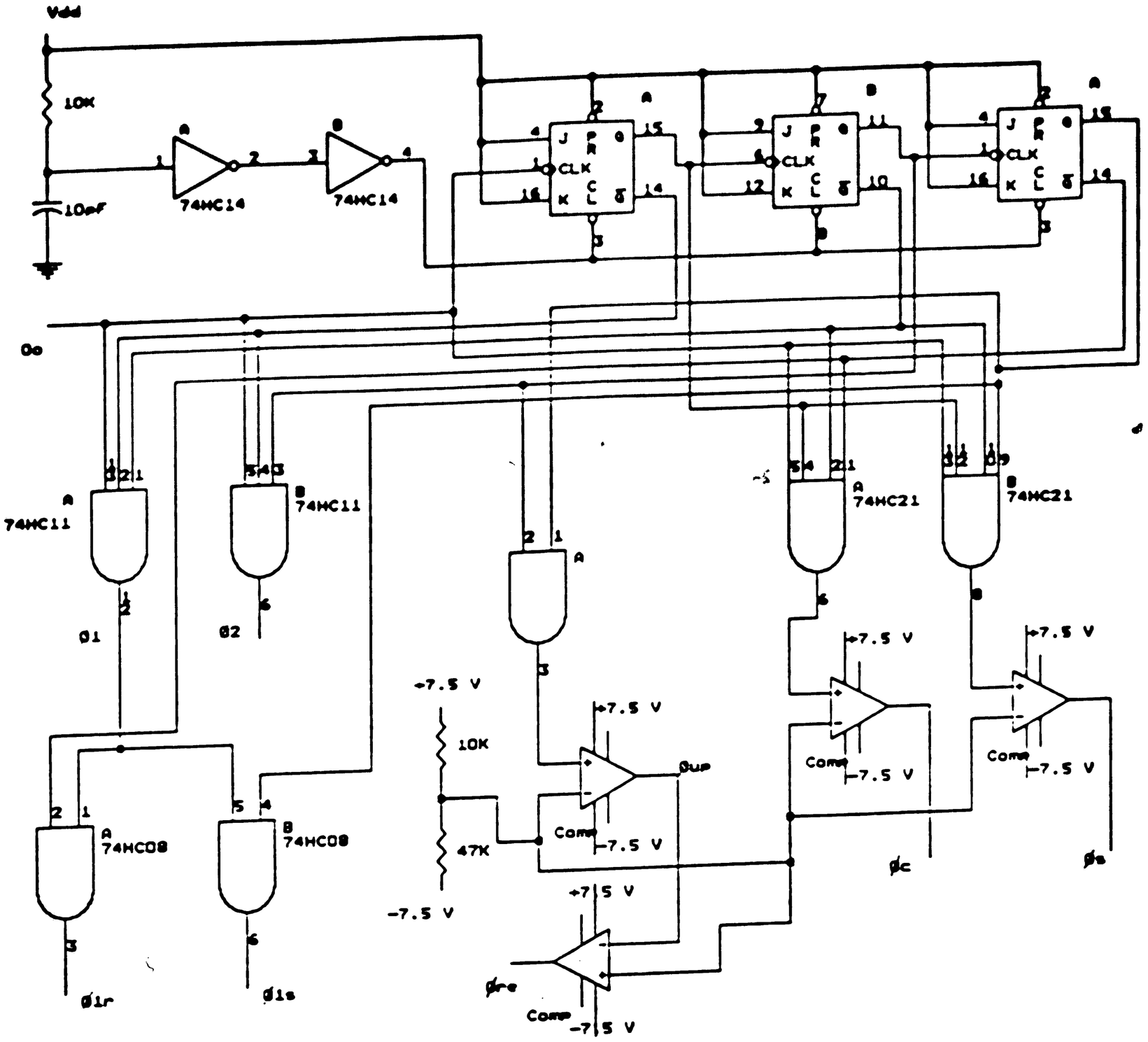


Figure 5-3: Schematic of the Digital Control/Clocking Module

frequency. Since we rely on an accurate knowledge of the phase relationship between the tapped signals x_0 and x_1 (i.e. they must be delayed to give $\omega_s T = \{[(2n+1)\pi/2]+\Delta\theta\}$ to an accuracy determined by $\Delta\theta$), we have to sample the input signal much faster than required by the Nyquist theorem. An error, $\Delta\theta$, in the phase relationship between x_0 and x_1, will determine the mean square error. For the linear adaptive neuron setup, the sampling clock is 16 times faster than

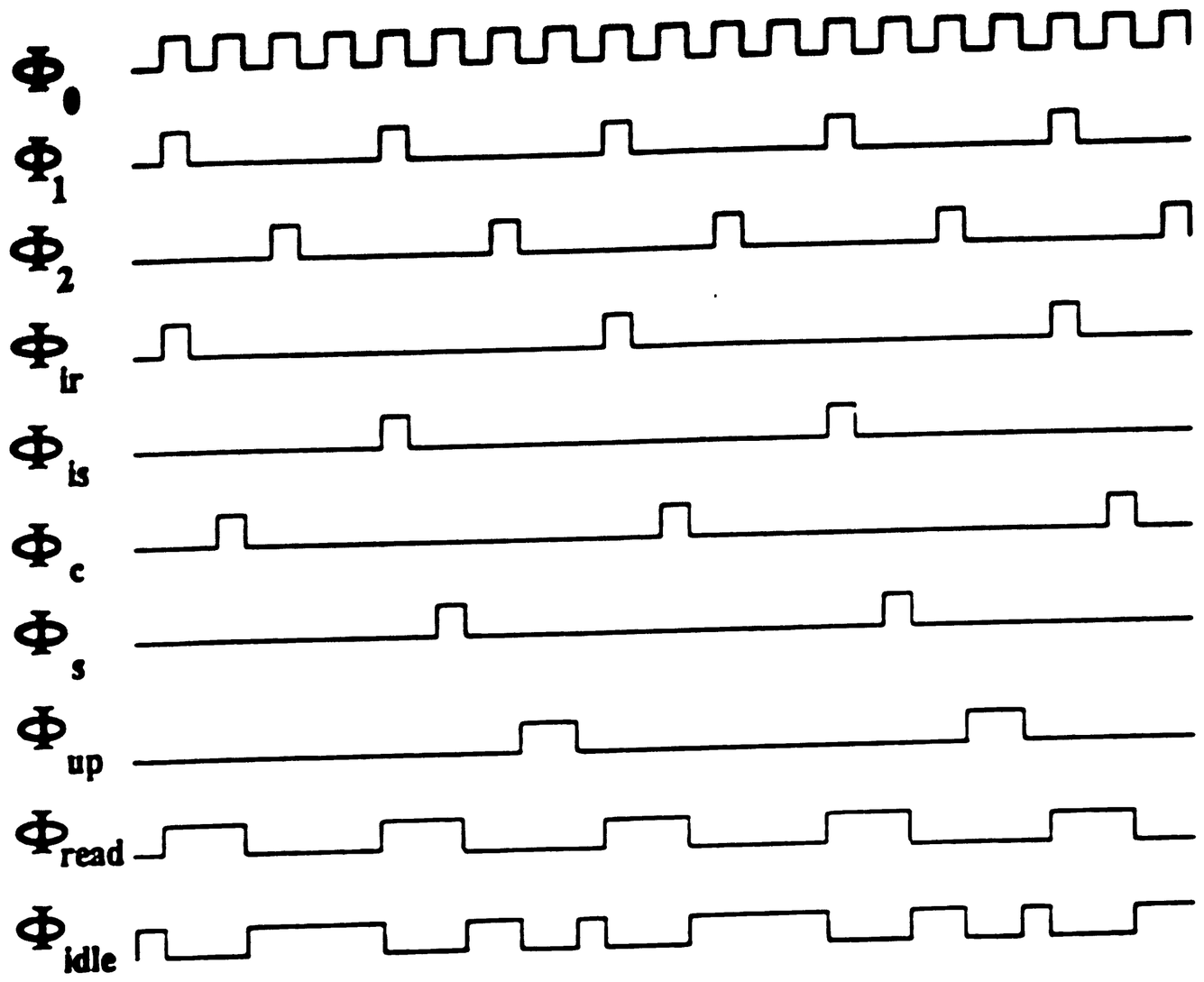


Figure 5-4: Timing Diagram of the Digital Control/Clocking Module with Idle Clock Signal

the input signal frequency (and therefore, Φ_1 and Φ_2 are 32 times faster than the input signal frequency).

5.1.2 The Analog Delay Line

The function of an analog delay line is to transfer the information (either as charge or voltage) through distinct but similar stages in the circuit. A main objective in this transfer processis to keep the information loss as small as possible. Common analog delay lines are implemented by Charge Coupled Device (CCD) or Bucket Brigade technologies. In the linear adaptive neuron

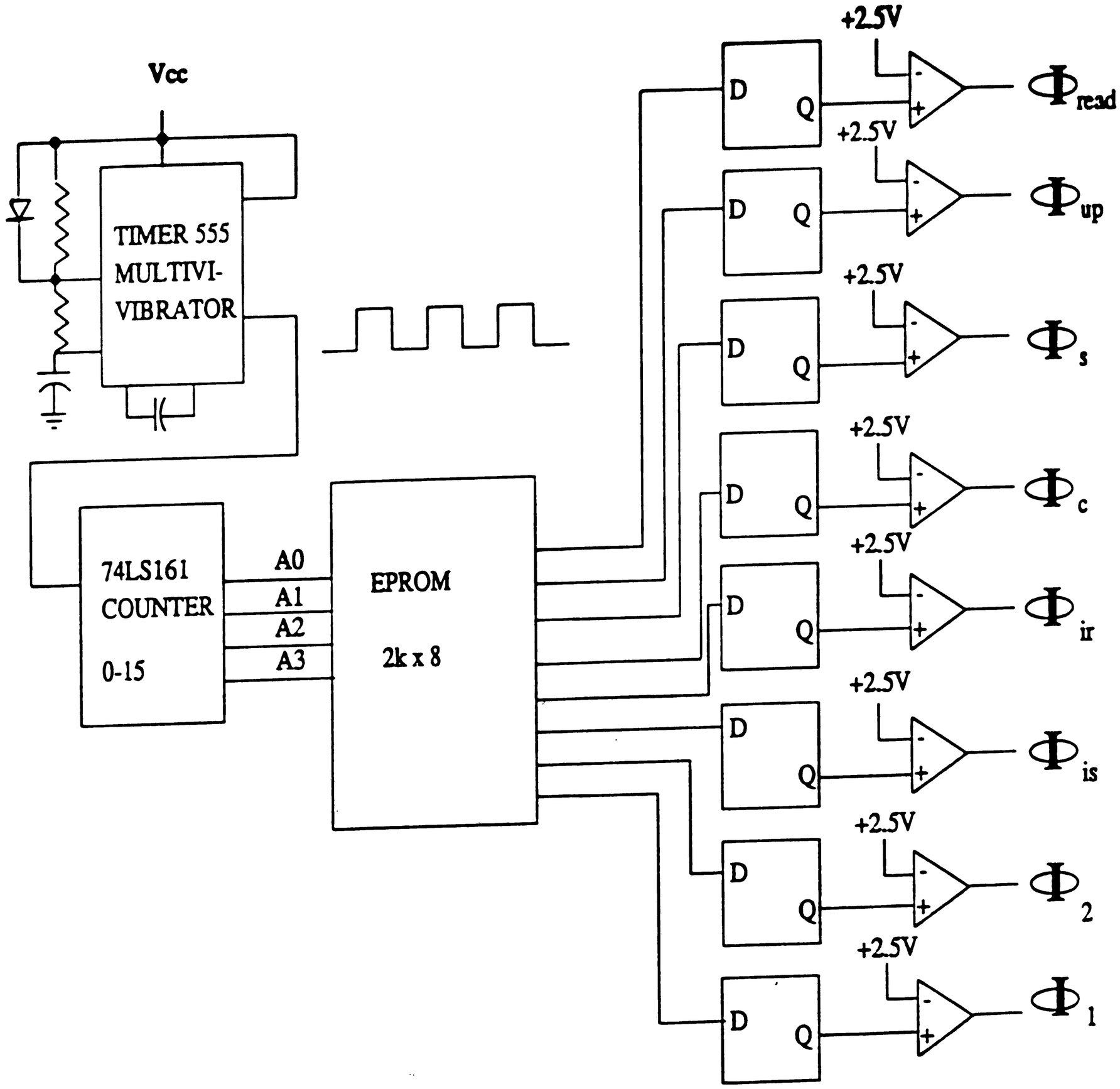


Figure 5-5: Schematic of the Digital Control/Clocking Module with Idle Clock Signal

circuit, the analog delay line is implemented in the switched capacitor fashion. The switched capacitor scheme is chosen because of the ease of incorporation of the analog delay line into the CMOS technology. Switched capacitor analog delay lines operate on the principle of transferring charges and storing the charge information on the storage capacitor. The clock controlled switches are responsible for charging and discharging the storage capacitors. The operation amplifier is used to provide the charging current for the storage capacitor.

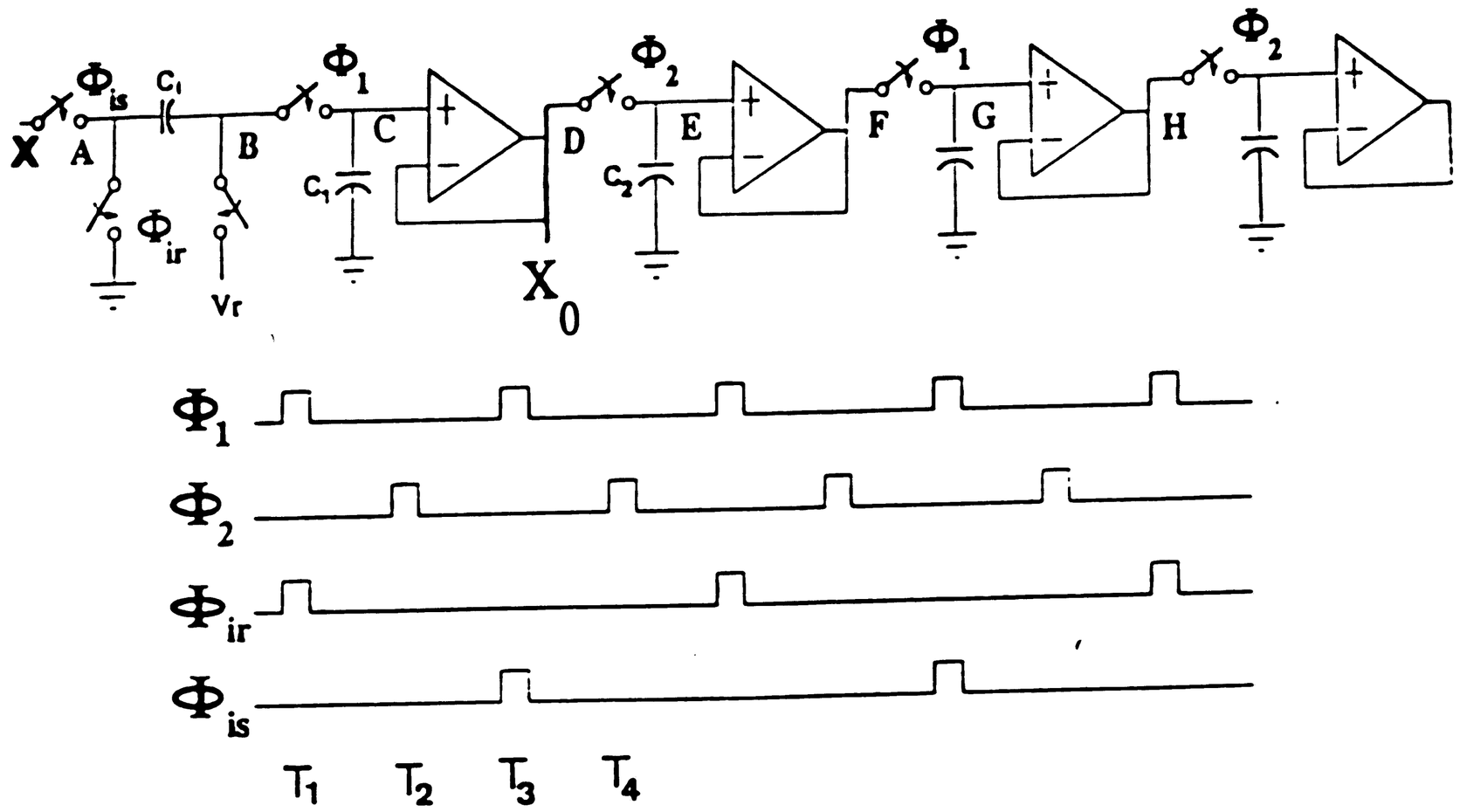


Figure 5-6: Schematic of the Analog Delay Line with Input Formation Circuitry

The analog delay line is tapped at different spatial locations to provide the delayed version of the input signal. In order to pass the input signal down the delay line, the tapped signal has to be read out nondestructively. The operational amplifiers used in the delay line also serve as buffers at these tapped locations. The speed of the delay line is determined by the controll clocking scheme described in the previous section. It is important to note that if

there are more stages between two adjacent tapped locations, then we can sample the input waveform in finer detail to obtain better information about the input signal. As mentioned in the previous section, the sampling clock frequency is 16 times faster than the input signal frequency (the analog delay line sampling clock, Φ_1 and Φ_2, are 32 times faster than the input signal frequency). Therefore, we can adjust the analog delay line sampling clock and/or the spatial tapped location to achieve 90 degree phase difference between two tapped signals, x_0 and x_1. Figure 5-6 shows the circuit schematic of the analog delay line with the input formation circuitry.

The clocking diagram of Φ_1,Φ_2,Φ_{ir}, and Φ_{is} are also shown in the figure in order to aid in the explanation of the circuit operation. Let us divide the time into different time frames, namely T_1, T_2, etc. and label reference points along the analog delay line as A, B, etc. During T_1, both Φ_1 and Φ_{ir} is high, and the voltage V_{ref} appears across the capacitors C_1 and C_i. During the time T_2, the voltage on the capacitor C_1 is transferred to capacitor C_2. When the time progresses to T_3, Φ_{is} is high and thus point A has the same potential as the input voltage. Since the capacitor holds the reference voltage V_{ref} across it and the voltage on a capacitor can not change instantaneously, point B has the potential of the sum of the input signal voltage and the reference voltage. Therefore, if we concentrates on one reference point, say point C, we will observe the alternation of reference voltage and the sum of input signal voltage and the reference voltage as the time progress from one frame to the next. Table 5-3 shows a good summary of the analog delay line operation at different time frame and at each reference point.

It is necessary to choose the capacitance value in the analog delay line to ensure the proper signal transfer. Several criteria have to be imposed in order to choose the appropriate capacitance value. First, the capacitance value has to

Analog Delay Line Operation		
Time Frame	Reference Point	Voltage Level
T_1	A	Ground
	B, C, D	V_{ref}
	E, F	$V_{ref} + x$
	G, H	$V_{ref} + x$
T_2	A	Ground
	B, C, D	V_{ref}
	E, F	V_{ref}
	G, H	$V_{ref} + x$
T_3	A	x
	B, C, D	$V_{ref} + x$
	E, F	V_{ref}
	G, H	V_{ref}
T_4	A	x
	B, C, D	$V_{ref} + x$
	E, F	$V_{ref} + x$
	G, H	V_{ref}

Table 5-3: Analog Delay Line Operation

be large enough to hold the charge stored on the capacitor and thus increase signal transfer efficiency. If we define Δt as the time when the charging switch is open and I_{leak} is the leakage current associated with the period, then we can calculate the change or loss of the capacitor voltage (ΔV)as shown in the following expression.

$$I_{leak} \cdot \Delta t = \Delta V \cdot C \tag{5.1}$$

where C is the value of the storage capacitor. However, if the storage capacitance is too large, then it will result in long charging RC time constant causing the storage capacitor to undercharge during the short charging time

allowed by the clock signal. It is also important to note that a large capacitance yields a long discharging time constant which is undesirable. These contradicting criteria must be satisfied simultaneously in order to choose proper capacitance value. The leakage current and ON resistance can be obtained from the data book for 4066 analog switch and thus 0.1 μF is determined to perform well in my experimental setup.

5.1.3 The Analog Signal Processor

The tapped signal from the analog delay line, x_0 and x_1, and the desired signal, d, serve as the input signals in this module while the output signal is the error signal, ε. Figure 5-7 shows the circuit schematic of the analog signal processor. The input signal, x_0, is connected to the drain terminals of a pair of SONOS nonvolatile transistors served as a weight element. The drain current of each SONOS transistor is the product of the input signal and the channel conductance. The source terminal of each transistor is connected to a summing path where all the drain current is summed. In the linear adaptive neuron setup, two summing paths are provided for each pair of the SONOS transistors. The upper summing path is defined as the positive summing path and the lower summing path is identified as the negative summing path. These two summing paths will eventually combine to form a single ended output and feed into a correlated double sampling circuitry to removed the unwanted noise and the offset voltages introduced by the operational amplifiers and the switches. The output of the neuron, y, is then compared with the desired or training signal, d, to generate the error signal, ε, which is to be minimized in order to train the system.

Let us define the conductances which are connected to the positive summing path with a superscript of + and the conductances connected to the

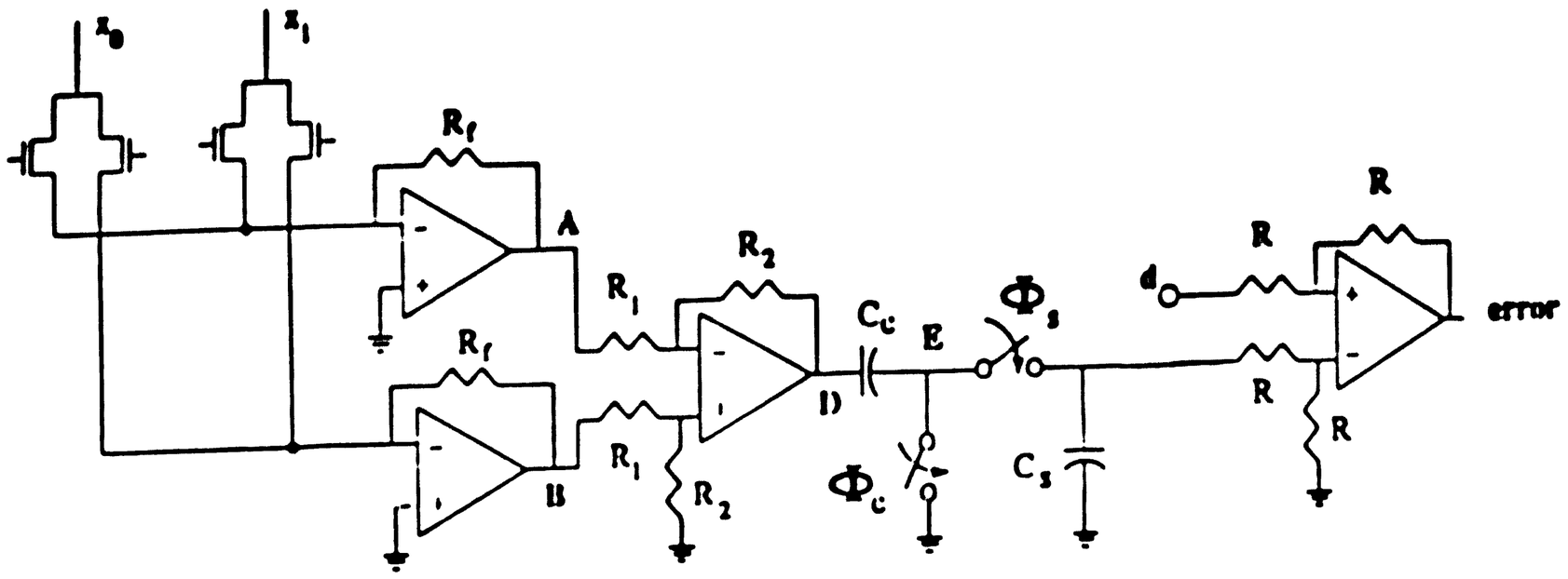


Figure 5-7: Schematic of the Analog Signal Processor

negative summing path with a superscript of -. First, we will focus our analysis on the positive summing path. The input signal (x_0) multiplied by the conductance of the left transistor in the transistor pair (g_{ds0}^{+}) is combined with the product of the input signal (x_1) and the conductance (g_{ds1}^{+}) in the positive summing path and is converted into a corresponding voltage by the operational amplifier with a gain of $-R_f$. Therefore, we can express the voltage at point A of the differential amplifier for small V_{ds} as

$$V_A = -R_f \cdot [x_0 \cdot g_{ds0}^{+} + x_1 \cdot g_{ds1}^{+}] \tag{5.2}$$

On the other hand, the same input signals are multiplied with the conductances of the right transistors and voltage converted to yield the voltage at point B as

$$V_B = -R_f \cdot [x_0 \cdot g_{ds0}^{-} + x_1 \cdot g_{ds1}^{-}] \tag{5.3}$$

Since we require a single ended output in order to generate error signal. V_A and V_B will be summed by the differential amplifier to yield the output y_n as

$$y_n = -(R_2/R_1)\cdot(-R_f)[x_0\cdot g_{ds0}{}^{+} + x_1\cdot g_{ds1}{}^{+}] + [(R_2/R_1)]\cdot(-R_f)[x_0\cdot g_{ds0}{}^{-} + x_1\cdot g_{ds1}{}^{-}]$$
$$= \frac{R_f R_2}{R_1}[x_0\cdot(g_{ds0}{}^{+} - g_{ds0}{}^{-}) + x_1\cdot(g_{ds1}{}^{+} - g_{ds1}{}^{-})] \qquad (5.4)$$

If we define the output y_n to be the linear product sum between the input signal and the weight value such as

$$y_n = x_0\cdot W_0 + x_1\cdot W_1$$

then we can define the weight value to be the difference in conductance value in each of the transistor pair.

$$W_0 = (\frac{R_f R_2}{R_1})\cdot(g_{ds0}{}^{+} - g_{ds0}{}^{-}) = (\frac{R_f R_2}{R_1})\beta_{eff}(V_{th0}{}^{-} - V_{th0}{}^{+})$$
$$W_1 = (\frac{R_f R_2}{R_1})\cdot(g_{ds1}{}^{+} - g_{ds1}{}^{-}) = (\frac{R_f R_2}{R_1})\beta_{eff}(V_{th1}{}^{-} - V_{th1}{}^{+}) \qquad (5.5)$$

A more exact analysis which includes second order V_{ds} nonlinear terms in I_{ds} gives the sdsame expressions for the weight values when correlated double sampling is employed. From this point on, the quantity $(R_f R_2)/R_1$ will be referred as the gain factor of the differential amplifier section. Therefore, the weight value can be either positive or negative as discussed in the previous chapter.

The weighted sum of the input signals, y_n, contains the DC reference voltage and noise which are also amplified by the differential amplifier. The correlated double sampling circuit is designed to removed all the common signal between each clamping and sampling cycle. The clamping clock, Φ_c, goes high immediately after the Φ_1 and Φ_{ir} clocks phase high. Both the input terminals,

x_0 and x_1, are at the DC reference voltage in potential. Therefore, point D at one end of the clamping capacitor, C_c, will be charged to the gain factor multiplies the reference voltage and the summation of two weight value while the other plate of the capacitor (point E) is grounded. During the next time frame, the sampling clock Φ_s, goes high and the input terminals contain the summation of the DC reference voltage and the AC input signal. The input signals are amplified by the differential amplifier network and appear at point D. Since the voltage across the clamping capacitor can not be changed instantaneously, the voltage at point E will become the gain factor multiplies the product of the pure AC input signals and their prospective weight values. The correlated double sampling operation can be best explained in Table 5-4. Notice, the 'clean' output of the correlated double sampling circuit is valid after the clock Φ_s is high. Once the output signal is valid, it is compared with the sampled version of the desired (training) signal and the difference is computed by the differential amplifier to generate the error signal.

Correlated Double Sampling Circuit Operation		
Time Frame	Reference Point	Voltage Level
Clamping	D	$\frac{R_f R_2}{R_1}[V_{ref} \cdot W_0 + V_{ref} \cdot W_1]$
	E	Ground
Sampling	D	$\frac{R_f R_2}{R_1}[(V_{ref}+x_0) \cdot W_0 + (V_{ref}+x_1) \cdot W_1]$
	E	$\frac{R_f R_2}{R_1}[x_0 \cdot W_0 + x_1 \cdot W_1]$

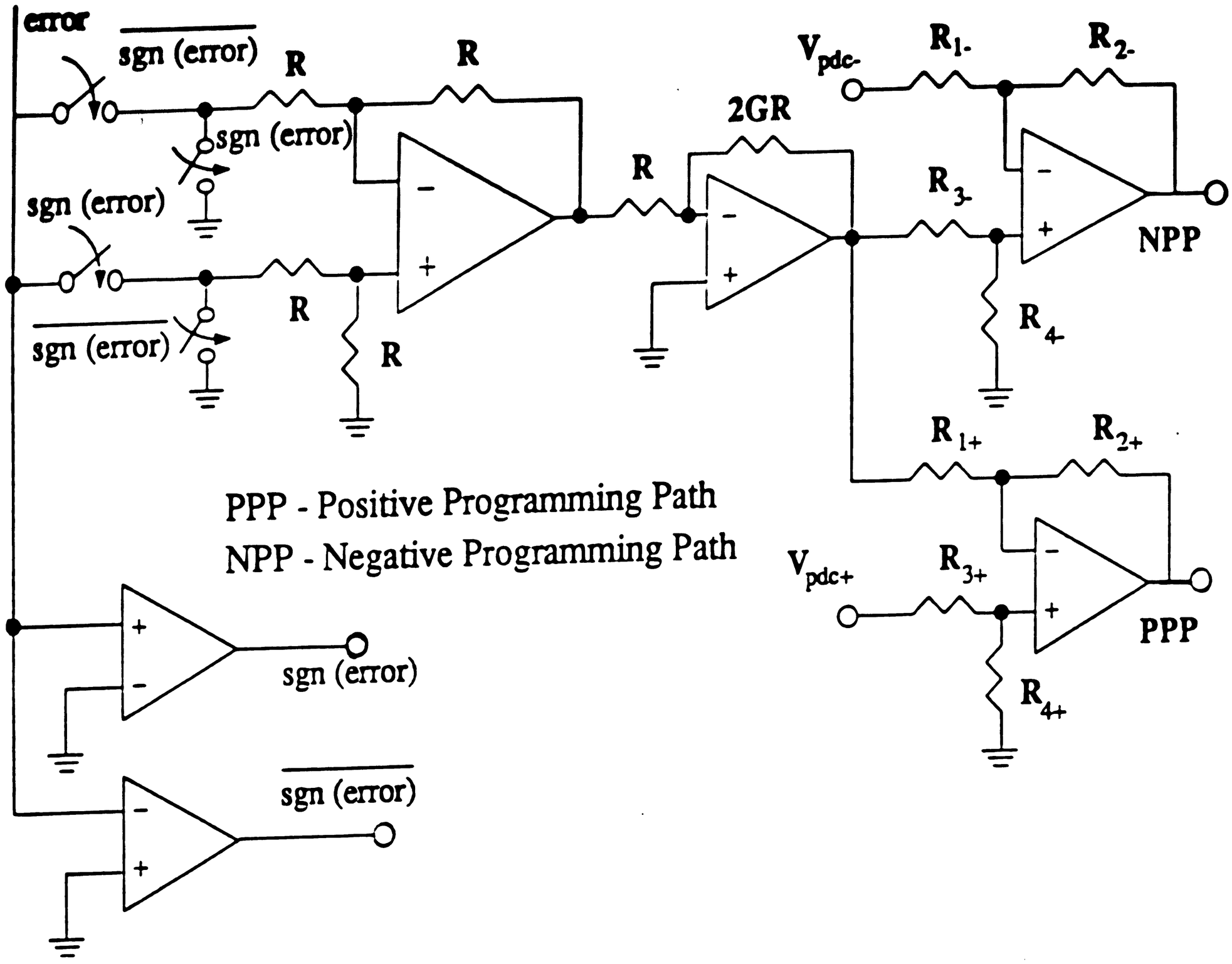
Table 5-4: Correlated Double Sampling Circuit Operation

5.1.4 The Learning Algorithm Module

The primary function of this module is to calculate the incremental weight value such that the overall error signal can be minimized in an iterative manner. Once the error is generated and fed from the analog processor module, the learning algorithm module takes the magnitude and sign information of the error and outputs two programming paths. The positive programming path amplifies the error magnitude by 2G times. The gain of the module, G, has the range from 0 to 1. On the other hand, the negative programming path amplifies the error magnitude by -2G times. Each programming path represents the positive or negative voltage used as the programming voltage to the gate of the SONOS transistors during update mode. The programming paths are level shifted to add a DC bias to the amplified AC error signal in order to program the SONOS transistor more effectively.

The circuit schematic of the learning algorithm module is shown in figure 5-8. The magnitude information of the error signal can be obtained by passing the error signal into an absolute value extractor as shown in the input section of figure 5-8. First, the sign information of the error signal is extracted by using a comparator with inverting terminal connected to ground. The sign information is then inverted and both the sign and the inverted sign signals are used to drive the switches in the absolute value extractor. If the error signal is determined to be positive, the switches configure the operational amplifier as noninverting amplifier and, thus, the error signal comes out unchanged. On the other hand, if the error signal is determined to be negative, the switches then set the operational amplifier as an inverting amplifier and alter the incoming error signal. The sign information is also used in the algorithm to calculate the incremental weight.

The learning algorithm implemented in the linear adaptive neuron is the



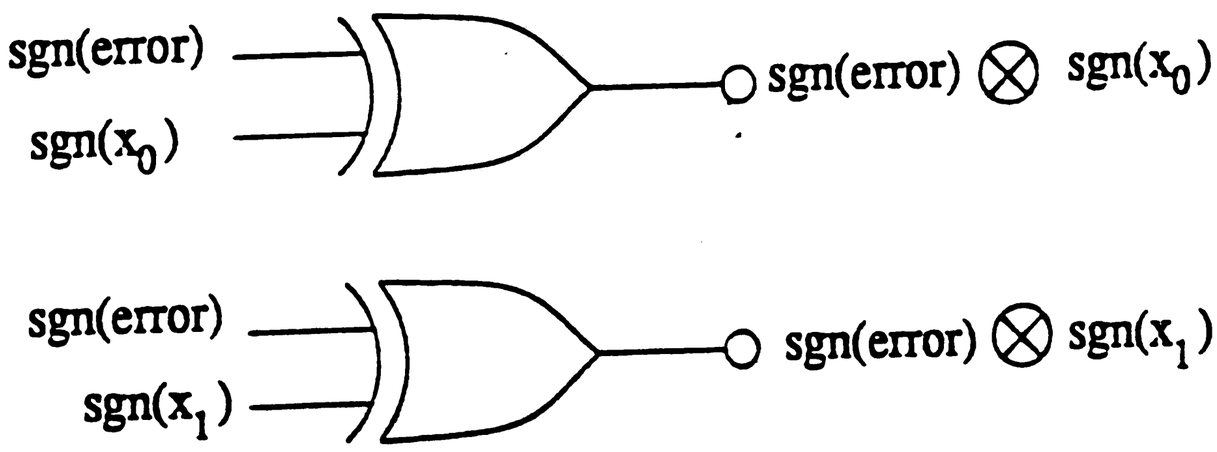


Figure 5-8: Schematic of the Learning Algorithm Module

clipped data algorithm discussed in chapter 2 and 4. The incremental weight can be written as

$$\Delta W_k = 2\mu|\varepsilon| sgn(\varepsilon)\cdot sgn(x_k)$$

where index k denotes the spatial location of the weight. If we assign logic levels 1 to represent the sign information of a positive error signal and logic 0 to represent the sign of a negative error signal, then the operation $sgn(\varepsilon)\cdot sgn(x_k)$ can be considered as a digital multiplication which can be implemented by a simple Exclusive OR operation. According to the result of the Exclusive OR, the steering network described in the following section will take either the positive programming path voltage or the negative programming path voltage and direct it to the gate of the SONOS memory transistor during the update mode.

5.1.5 The Steering Network

The steering network is composed of switches controlled by the output from the learning algorithm module to steer proper programming voltages from the programming paths. In order to program the SONOS weight element more effectively, we program the channel conductances of each transistor pair in the opposite direction. For example, if the algorithm require an increase in the overall weight value, then the channel conductance of the left transistor, in the transistor pair, (g_{ds}^{+}), will be increased by having the negative programming path connected to the gate terminal (erase operation) while the channel conductance of the right transistor, (g_{ds}^{-}), will be decreased by the programming voltage from the positive programming path (write operation). This scheme is chosen because of the faster convergence in the weight value. Figure 5-9 shows the circuit schematic of the steering network. If g_{ds0}^{+} denotes the left transistor of W_0 pair and g_{ds0}^{-} denotes the right transistor of W_0 pair, then we can

summarize the steering network operation (programming voltage) in Table 5-5. The steering network is also responsible for switching in the read voltage during

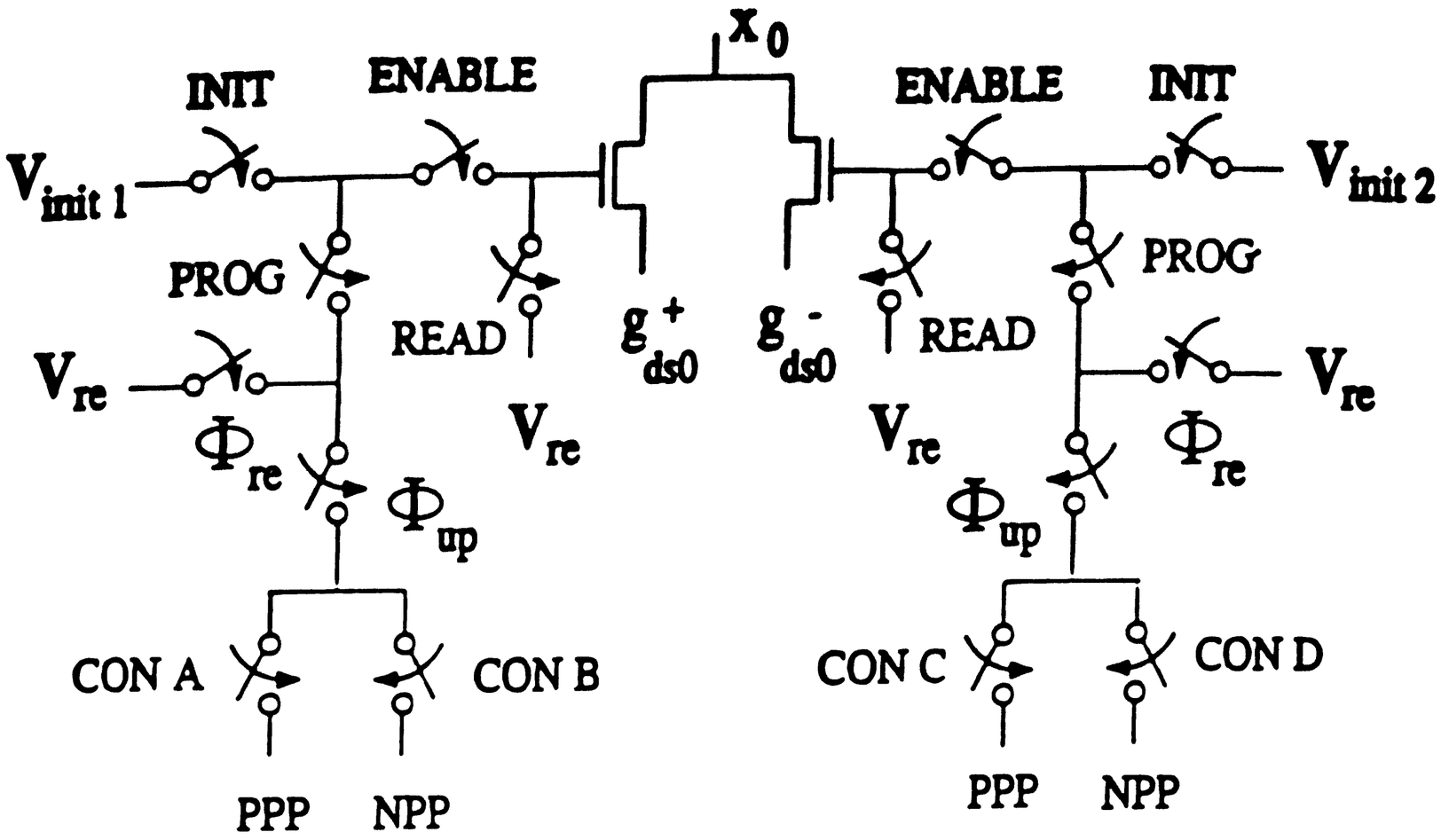
Steering Network Circuit Operation					
$sgn\varepsilon sgnx_0$	$sgn\varepsilon sgnx_1$	g_{ds0}^{+}	g_{ds0}^{-}	g_{ds1}^{+}	g_{ds1}^{-}
+	+	$-2G\|\varepsilon\|$	$+2G\|\varepsilon\|$	$-2G\|\varepsilon\|$	$+2G\|\varepsilon\|$
+	-	$-2G\|\varepsilon\|$	$+2G\|\varepsilon\|$	$+2G\|\varepsilon\|$	$-2G\|\varepsilon\|$
-	+	$+2G\|\varepsilon\|$	$-2G\|\varepsilon\|$	$-2G\|\varepsilon\|$	$+2G\|\varepsilon\|$
-	-	$+2G\|\varepsilon\|$	$-2G\|\varepsilon\|$	$+2G\|\varepsilon\|$	$-2G\|\varepsilon\|$

Table 5-5: Steering Network Circuit Operation

the read mode, the initializing voltages during the initializing mode, and the read voltage during the disabled mode, to the gate terminals of the SONOS transistors to ensure proper operation of the entire circuit.

5.2 Measurement Setup and Results

The electrical performance of the linear adaptive neuron is demonstrated in two major characteristics: (1) The output and training signals versus time, and (2) The error signal versus time. The former characteristics provides the information of how well the output signal approximates the training signal especially in the phase relationship between these two signals. The latter characteristics shows how fast the linear adaptive neuron adapts before it reaches its minimum error. All the waveforms and photographs are taken from a Tektronix 7854 Digital Storage Oscilloscope. The power supplies not only supply the power rails for circuit operation, but they also provide a convenient way for adjusting the variable reading, DC reference, positive programming shift, and negative programming shift voltages. The function generator is used to provide the input and training signals. The training signal amplitude is set

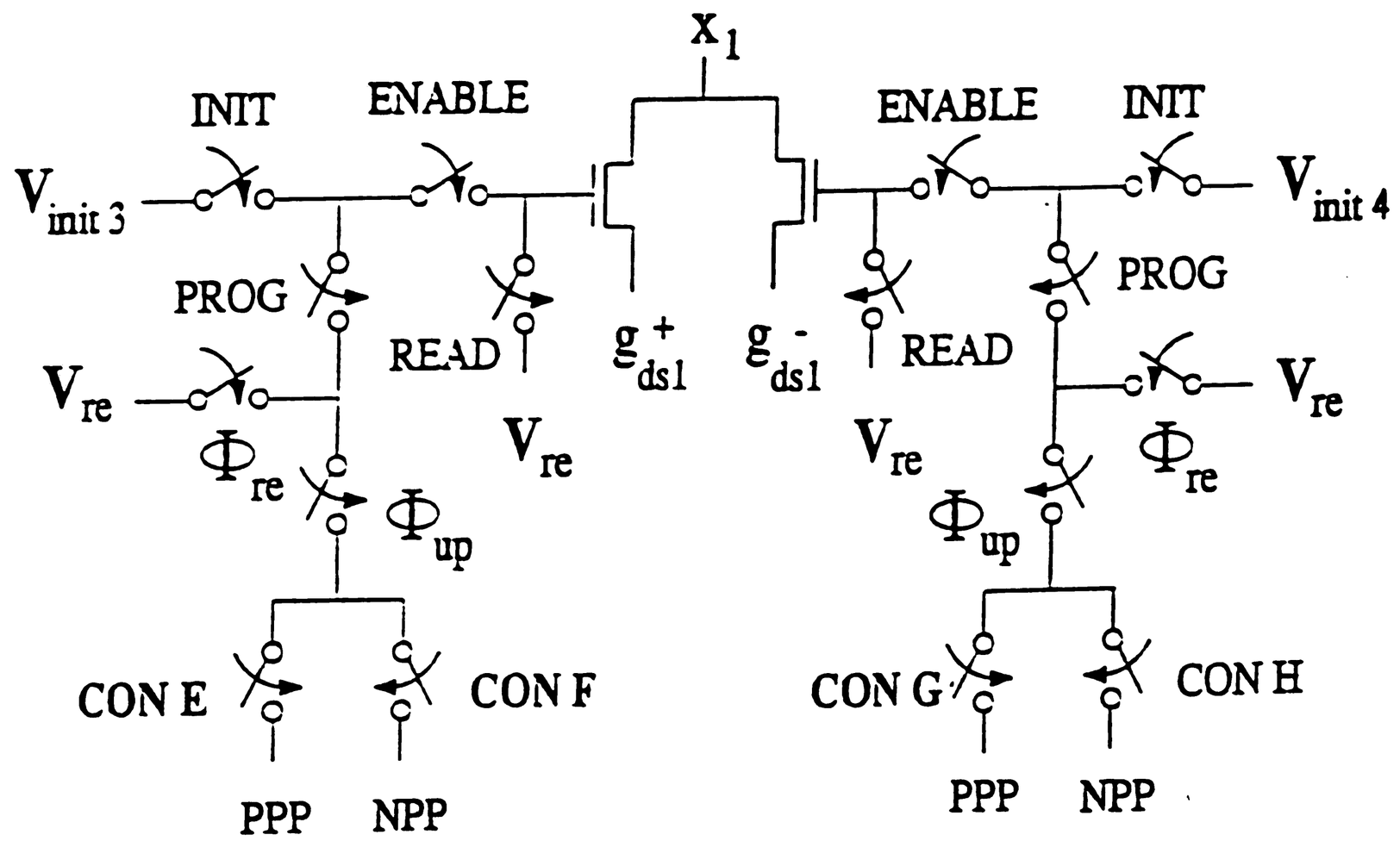


sgn (error) sgn(x_0) ==> Positive

CON A = LOW CON C = HIGH
CON B = HIGH CON D = LOW

sgn (error) sgn(x_0) ==> Negative

CON A = HIGH CON C = LOW
CON B = LOW CON D = HIGH



sgn (error) sgn(x_1) ==> Positive

CON E = LOW CON G = HIGH
CON F = HIGH CON H = LOW

sgn (error) sgn(x_1) ==> Negative

CON E = HIGH CON G = LOW
CON F = LOW CON H = HIGH

Figure 5-9: Schematic of the Steering Network

up to be 10 times greater than the input signal. Furthermore, the training signal is phase-shifted by 45 degrees with respect to the input signal in order to demonstrate both the amplitude and phase adaptation of the linear adaptive neuron. There exists a 90 degree phase shift between the two tapped input signals as described and analyzed in section 5.1.

5.2.1 Output and Training Signals versus Time Characteristics

The output and training signals versus time characteristics consist of two parts: the initialized and the adapted part. In the initialized part, the weights are initialized to a known state (either the fully positive state or the fully negative state). The fully positive state is achieved by applying a negative programming gate bias to the transistors connected to the positive programming path and applying a positive programming gate bias to the transistors connected to the negative programming path. To obtain the fully negative state in weight value, the programming gate biases applied to the channel conductances are in the exactly opposite way.

The circuit is then placed under the disabled mode with the output and training signals monitored by the oscilloscope. Since the input and training signal have an amplitude and phase difference between them, the output signal (configured to represent the weighted sum of the input signals) and the training signal will have an initial amplitude and phase mismatch. Initially, the amplitude of the output signal is larger than the training signal because of the built-in gain of the circuit. However, if the circuit is placed under the programmed mode, then the output signal will shrink in amplitude and phase locked with the training signal as the error signal is minimized. Figures 5-10 a and b shows the output and training signals versus time characteristics before and after the adaptation takes place.

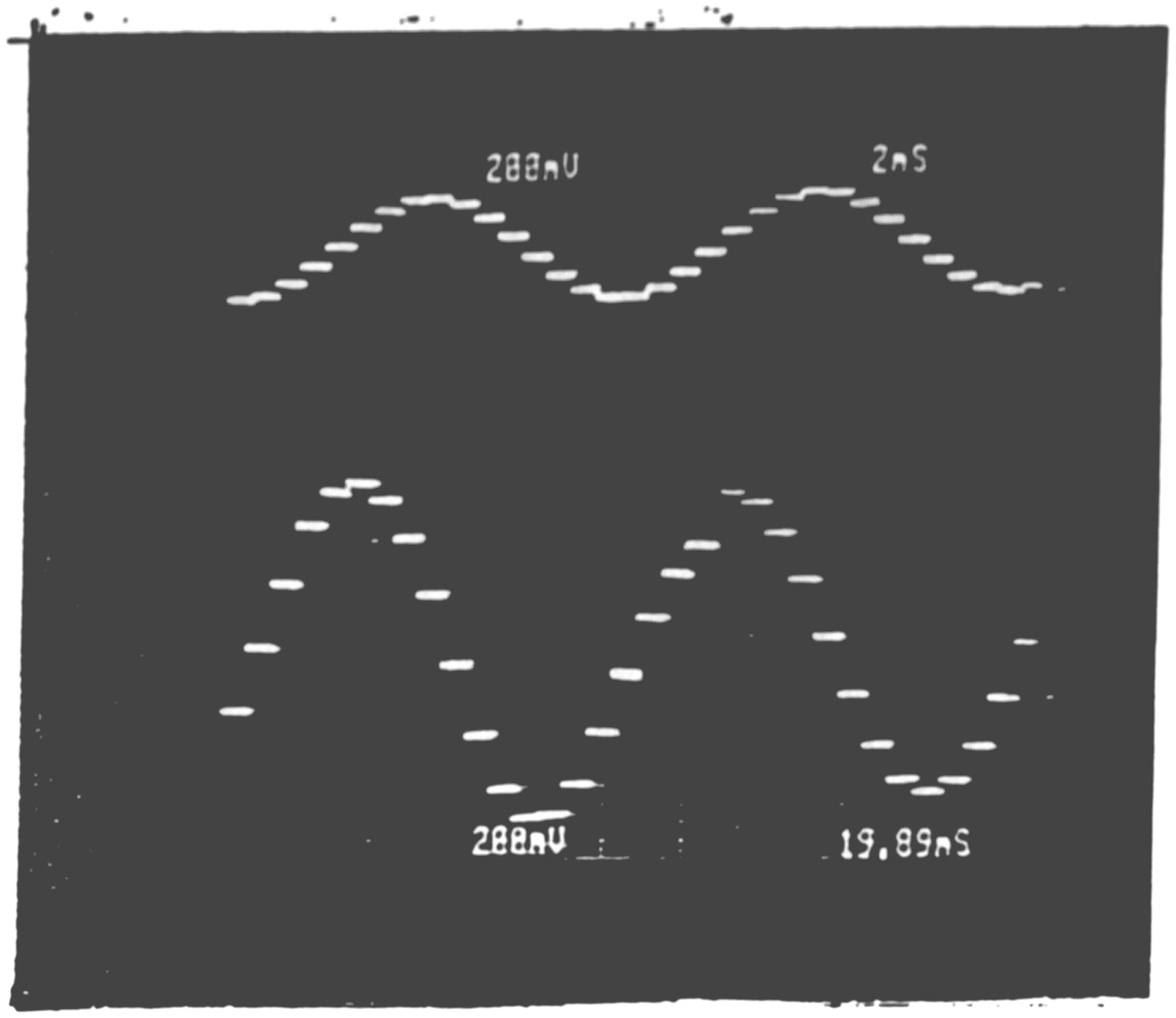


Figure 5-10: Output and Training Signals versus Time Characteristics: (a)Initialized and (b) Adapted

5.2.2 Error Signal versus Time Characteristics

The error signal versus time characteristics provides a quantitative information of how fast the linear adaptive neuron adapts to the training signal (when the error signal is at its minimum). Again the circuit has to start from known weight states. However, there exists a distinct difference in measurement technique between the error signal versus time characteristics and the output and training signal versus time characteristics. First of all, the circuit does not have to go through the disabled mode (in which the error feedback path is disconnected from the gate terminals of the SONOS devices and a read voltage is applied to all the gate electrodes of the SONOS devices to read out the conductance information) for the error signal versus time characteristics. Secondly, the digital oscilloscope is configured to perform a single sweep, one-shot triggering time base instead of the 'free-running' automatic time base - the normal operational mode for the oscilloscope. The oscilloscope is triggered to start the single sweep at the flip of the SPDT switch which switches the circuit from the initialized mode to the programmed mode. The oscilloscope continuously monitors the error signal, digitizes the data, and stores the data in the memory for further display. Once the time base limit is reached, the oscilloscope stops sampling the error signal and display all the data points on the CRT screen for photographic capture or data transfer through its IEEE 488 bus to a HP 9836 computer.

It is interesting to note that different weight initialization scheme have different effects on how the circuit converges to a minimum error. One possible explanation is the SONOS transistors have different erase characteristics than write characteristics. Therefore, when the weights are initialized differently, there are times when the circuit does more erasing than writing and vice versa, and thus creates different error versus time characteristics. Figures 5-11, 5-12,

5-13, and 5-14, show the effects of different initialization scheme on the performance.

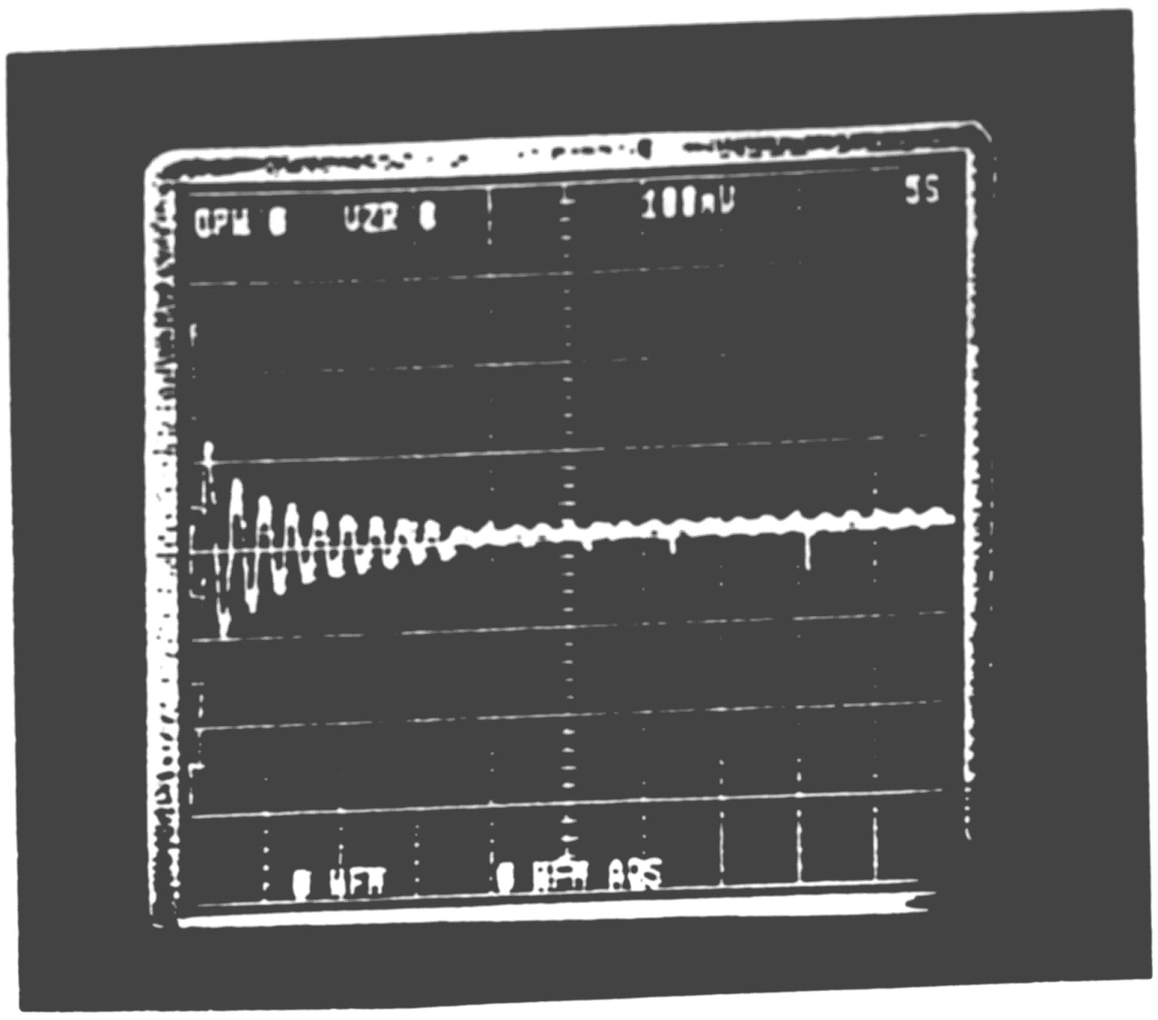


Figure 5-11: Error versus Time Characteristics - Initialization Scheme: + - + -

We will define the system time constant to be the time it takes for the error signal to decay from its maximum value to the 1/e of the maximum error signal. In addition, we can define the quantity of adaptivity as

$$Adaptivity = 20\ log_{10}\left(\frac{\varepsilon}{d_a}\right) \quad in\ steady\ state \tag{5.6}$$

The experimental results are summarized in Table 5-6. The adaptivity listed in Table 5-6 is calulated based on the final error amplitute at the end of the 50 second time span of adaptation. If the circuit is allowed to adapt for a longer period of time, then the adaptivity is found to be improved. From Table 5-6, the initialization scheme which has the smallest system time constant

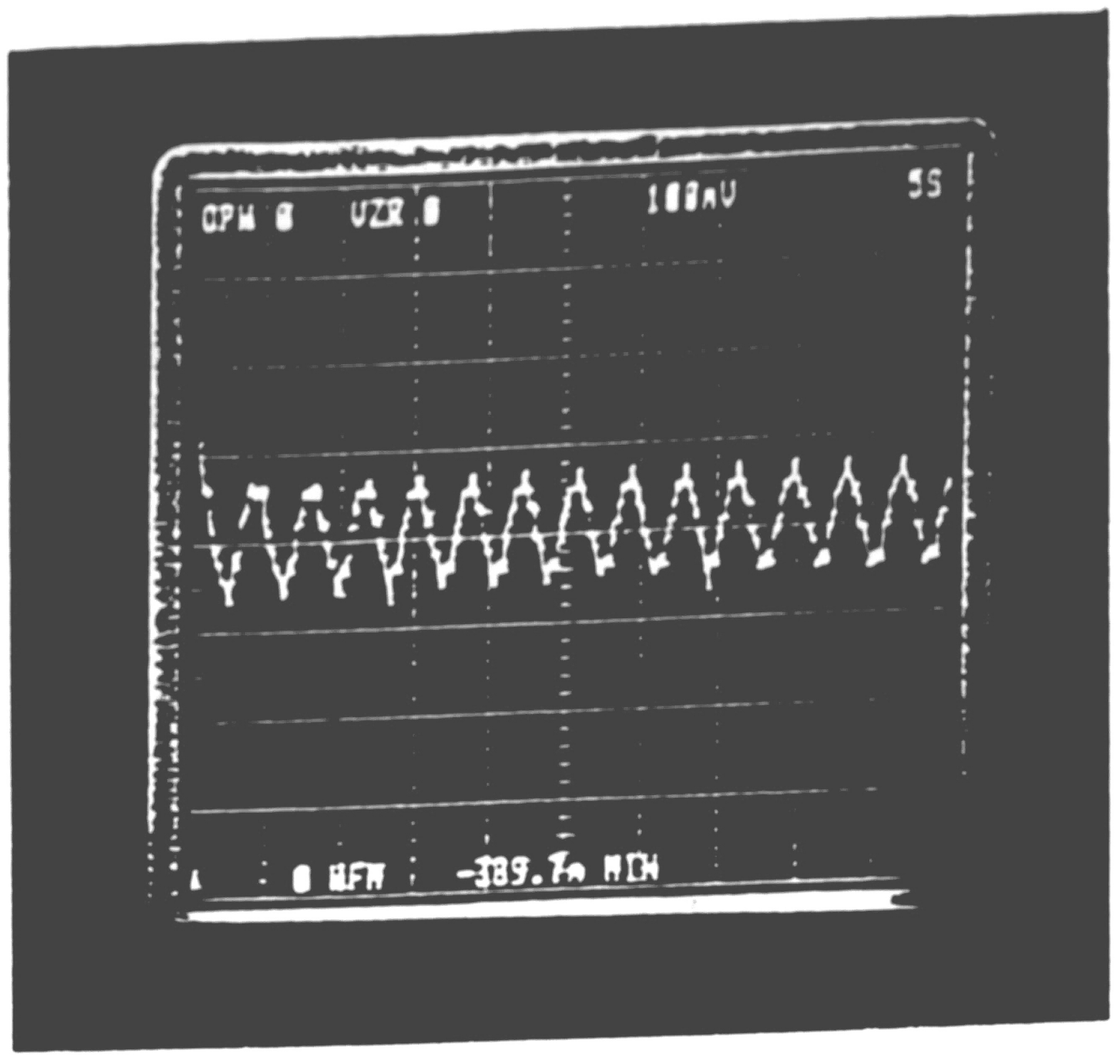


Figure 5-12: Error versus Time Characteristics - Initialization Scheme: + - - +

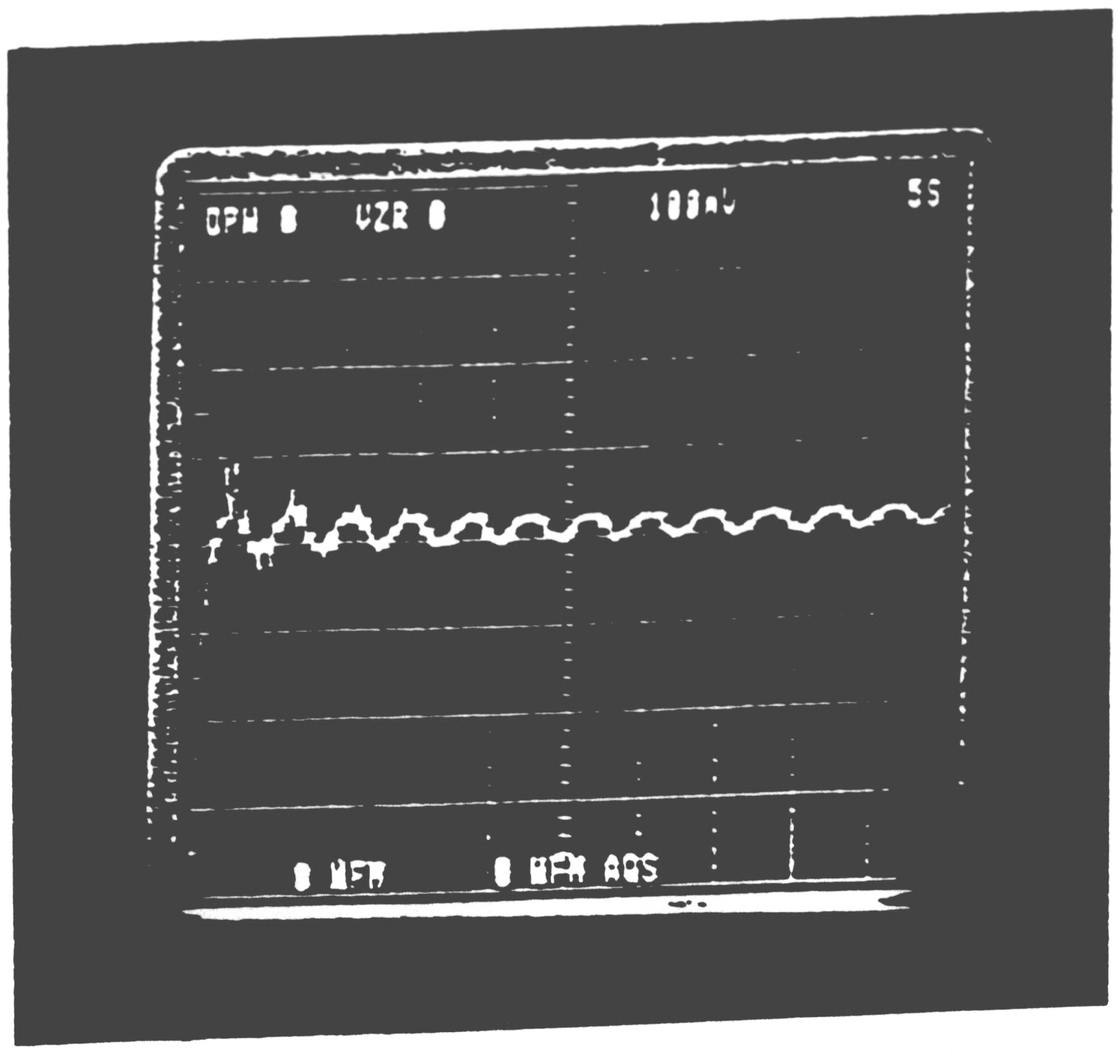


Figure 5-13: Error versus Time Characteristics - Initialization Scheme: - + + -

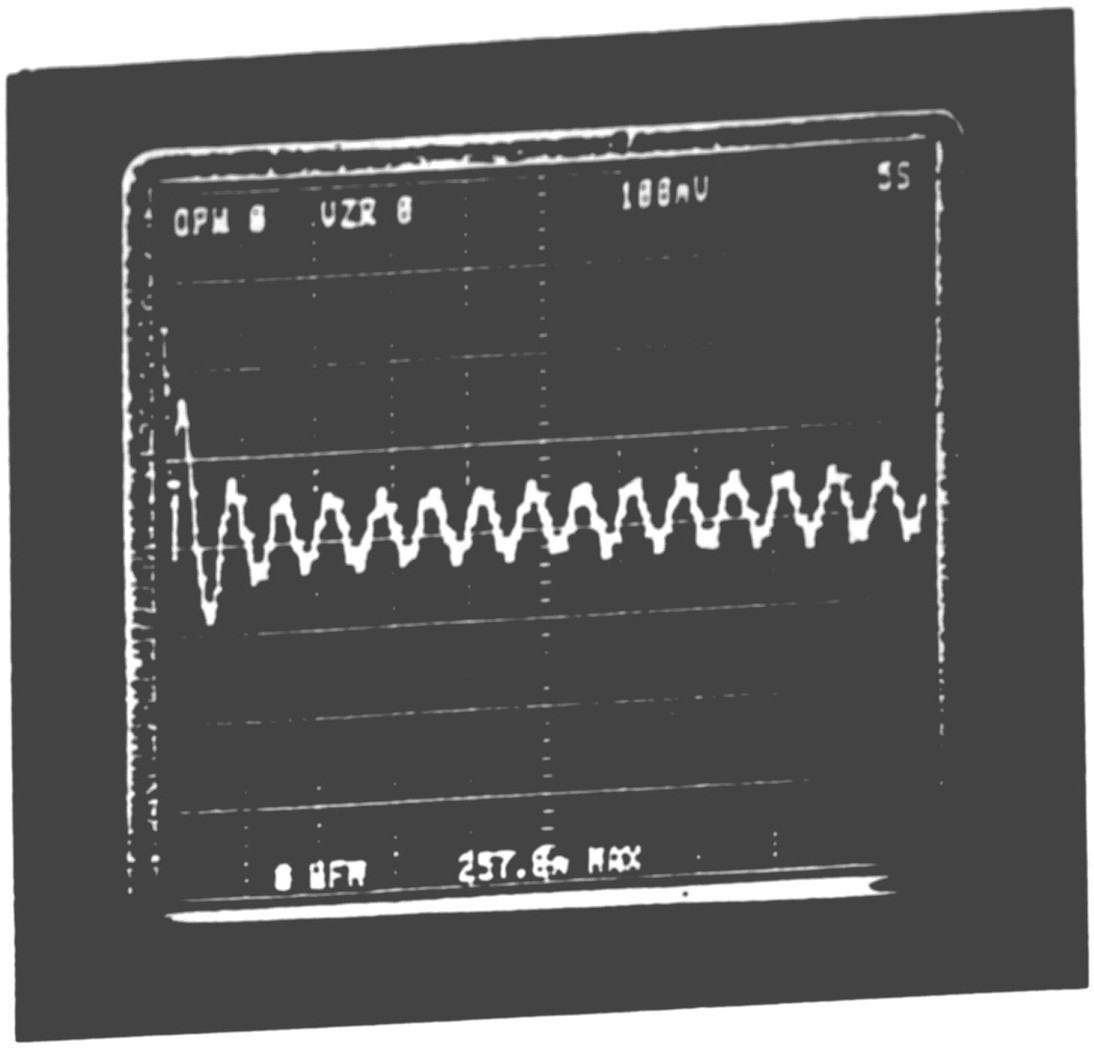


Figure 5-14: Error versus Time Characteristics - Initialization Scheme: - + - +

Summary of the Experimental Results		
Initialization	System Time Constant (s)	Adaptivity
+ - + -	3	-20
+ - - +	1	-4.4
- + + -	2	-16.4
- + - +	3	-7.9

Table 5-6: Summary of the Experimental Results

yields poorer adaptivity. More experiments have to be performed before any direct relationship between the system time constant and the adaptivity can be drawn.

Chapter 6
Conclusions

A solid state electronic two tap weight linear adaptive neuron has been designed and constructed in breadboard form. The operation theory of the linear adaptive neuron has also been explored in Chapter 4. In addition, the electrical performance and the experimental results of the linear adaptive neuron are presented in Chapter 5. The source code and the results of a software simulation, written to mimic the electrical neuron operation, is included in Appendix C. The two tap weight linear adaptive neuron with a Widrow-Hoff's delta learning rule serves as a test vehicle to demonstrate the salient features of the SONOS nonvolatile memory transistors used as the synaptic elements in the hardware implementation of neural networks. The SONOS fabrication technology and the characterization techniques of the SONOS devices are outlined in Chapter 3. These characterization techniques include the high frequency C-V characteristics, the linear voltage ramp characteristics, the erase/write characteristics, the retention characterisitcs and the dynamic range characterisitcs. The attractive features of the SONOS electrically modifiable synaptic elements are:

- Analog (Free from quantization error)
- Small Size (Estimated 20 μm^2/weight cell for 1.2 μm feature size)
- Low Programming Voltages (< 7.5 V)
- Low Power Consumption (< 1μW/weight cell)
- Good Dynamic Range (60 dB)
- Good Memory Retention (20% window at a projected 10 year period)

- Reinforced Learning
- Excitatory/Inhibitory Synaptic Behavior

The electrical performance of the linear adaptive neuron is found to be dependent on the weight initialization. It is postulated this observation is due to the nonsymmetry of the erase and write operation of the SONOS devices. The DC voltage shift added to the programming voltage has its influence on the circuit convergence as well. The circuit performance is found to be timing dependent. If all the controlling signals are made twice as slow, then the error amplitude is found to be larger due to the longer programming and reading times, resulting in a 'soft write' operation during reading. A theoretical analysis on the learning algorithms and the convergence factor has been presented in Appendix B.

The analog delay line, which is designed to give a 90 degrees phase shift between the tapped locations, may have a 'phase error' between the two tap locations. In the situation where no noise is present in the circuit, the theoretical analysis predicts the error signal becomes sinusoidal due to the phase error in the steady state condition. This prediction is confirmed by the experimental results shown in Chapter 5. The convergence time constant of the system is found to be in the seconds range, corresponding to several thousand of programming cycles. This observation indicates that the SONOS nonvolatile memory transistors are fairly insensitive to the short programming pulse duration (156 μs) used in the circuit. The adaptivity figures presented in Chapter 5 are based on 50 seconds of adaptation time. The adaptivity is found to be much better if the circuit is allowed to adapt for a long period of time, say 1 hour. The typical adaptivity after a long adaptation time is around -26dB. Although experimental results presented in Chapter 5 indicate the weight

initialization schemes which provide better convergence time performance show poorer adaptivity behavior, more experiments have to be conducted to draw a direct relationship between the system convergence time constant and the adaptivity of the system.

The angular dependence on the adaptivity between the training signal and the input signal has not been fully investigated. It is important to optimize not only the circuit design but also the SONOS nonvolatile transistors as well. It is desirable to have SONOS devices which program faster, retain memory better, and have symmetric erase and write characteristics. The placement of the memory window is also an issue of concern. If the memory window is centered at the ground level, then the reading voltage can be at the ground level, which minimizes the soft write operation of the SONOS memory transistor. In order to minimize the noise introduced in the circuit, the offset free switched capacitor analog delay line may be implemented with the expense of a more elaborate clocking module.

In order to realize a larger neural network, the synaptic element must be readily integratable onto silicon wafers. Since the SONOS technology is fully compatible with the CMOS technology, the work presented in this thesis should provide a leap toward the realization of artificial neural networks.

References

1. Bernard Widrow, Study Director, *DARPA Neural Network Study, 10/87-2/88, Final Report*, Lincoln Laboratory, MIT, MA, 1988.

2. H.P. Graf and P. deVegvar, "A CMOS Implementation of a Neural Network Model", *Proceedings of Stanford Conference on Advanced Research in VLSI*, 1987, pp. .

3. Y. Tsividis and S. Satyanarayana, "Analog Circuits for Variable-Synapse Electronic Neural Networks", *Electronic Letters*, Vol. 23, No. 24, Nov 1987, pp. 1313-1314.

4. F.J. Kub, I.A. Mack, K.K. Moon, C.T. Yao and J.A. Modla, "Programmable Analog Synapses for Microelectronic Neural Networks Using a Hybrid Digital-Analog Approach", *IEEE Device Research Coference on Neural Networks*, 1988, pp. 24-27.

5. Mark Holler, Simon Tam, Hernan Castro, and Ronald Benson, "An Electrically Trainable Artificial Neural Network (ETANN) with 10240 'Floating Gate' synapses", *Proceedings of IJCNN*, 1989.

6. M.H. White, I.A. Mack, G.M. Borsuk, D.R.Lampe, and F.J. Kub, "Charge-Coupled Device (CCD) Adaptive Discrete Analog Signal Processing", *IEEE J. of Solid-State Circuits*, Vol. SC-14, 1979, pp. 132.

7. Marvin H. White and Chun-Yu Chen, "Electrically Modifiable Nonvolatile Synapses for Neural Networks", *Proceedings of IEEE International Symposium on Circuits and Systems*, 1989, pp. 1213-1216.

8. Richard P. Lippmann, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, Vol. 4, No. 2, April 1987, pp. 4-22.

9. J.E. Spencer, "Real-Time Applications of Neural Nets", *IEEE Trans. on Nuclear Science*, Vol. 36, No. 5, October 1989, pp. 1485-1489.

10. Hans P. Graf and Lawrence D. Jackel, "ANalog Electronic Neural Network Circuits", *IEEE Circuit and Devices Magizine*, Vol. 5, No. 4, July 1989, pp. 44-49.

11. B. Widrow and S.D.Sterns, *Adaptive Signal Processing*, Prentice-Hall, 1985.

12. B. Widrow, P.E. Mantey, L.J. Griffiths, and B.B. Goode, "Adaptive Antenna System", *Proceedings of IEEE*, Vol. 55, No. 12, December 1967, pp. 2143-2159.

13. B. Widrow and M. Hoff,Jr., "Adaptive Switching Circuits", *IRE WESCON Conv. Rec.,pt. 4*, 1960, pp. 96.

14. J.L. Moschner, "Adaptive Filter with Clipped Input Data", Tech. report, Standford Lab. Report, No. 6796-1, June 1970.

15. D. Hirsch and W. Widrow, "A Simple Adaptive Equalizer for Efficient Data Transmission", *IEEE Trans. Comm. Tech.*, Vol. Com-18, 1970, pp. 5.

16. Frank Robert Libsch, *Physics, Technology and Electrical Aspects of Scaled MONOS/SONOS Devices for Low Voltage Nonvolatile Semiconductor Memories (NVSMs),* PhD dissertation, Lehigh University, 1989.

17. Margaret Larson French, "Memory Window Studies of Nonvolatile Silicon-Oxide-Nitride-Oxide-Silicon (SONOS) Memory Devices", Master's thesis, Lehigh Univeristy, 1990.

18. Anirban Roy, "Retention, Endurance and Interface Traps in MONOS Memory Transistors", Master's thesis, Lehigh Univeristy, 1985.

19. Marvin H. White, Donald R. Lampe, Franklyn C. Blaha, and Ingham A. Mack, "Characterization of Surface Channel CCD Image Array at Low Light Levels", *IEEE J. of Solid-State Circuits*, Vol. SC-9, No. 1, February 1974, pp. 1-13.

20. John J. Hopfield and David W. Tank, "Computing with Neural Circuits: A Model", *Science*, Vol. 233, August 1986, pp. 625-633.

21. J. Davis, R. Newburgh, and E. Wegman, editors, *Neural Dynamics of Category Learning and Recongnition: Attention, Memory Consolidation, and Amnesia,* Brain Structure, Learning, and Memory, AAAS Symposium Series, 1986.

22. R. Rosenblatt, *Principles of Neurodynamics,* Spartan Books, 1959.

23. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, *Learning Internal Representations by Error Propagation,* MIT Press, 1986.

24. T. Kohonen, *Self-Organization and Associative Memory,* Springer-Verlag, 1984.

25. Marvin H. White and J. Ronald Cricchi, "Characterization of Thin-Oxide MNOS Memory Transistors", *IEEE Transaction on Electron Devices*, Vol. ED-19, No. 12, December 1972, pp. 1280-1288.

Appendix A
Learning Algorithms

There are several algorithms proposed to be used as the learning algorithms for the neural networks as classifier. Figure A-1[8] shows a taxonomy of six neural networks. Generally, there are two types of input signals, binary and the continuous-value (analog). Under each signal category, it can be subdivided into two training methods, supervised or unsupervised. Those nets trained with supervised input can be best used as associative memories or as classifier. Those nets trained without supervision are used as vector quantizer. These six algorithm have been summarized and described in the following sessions.

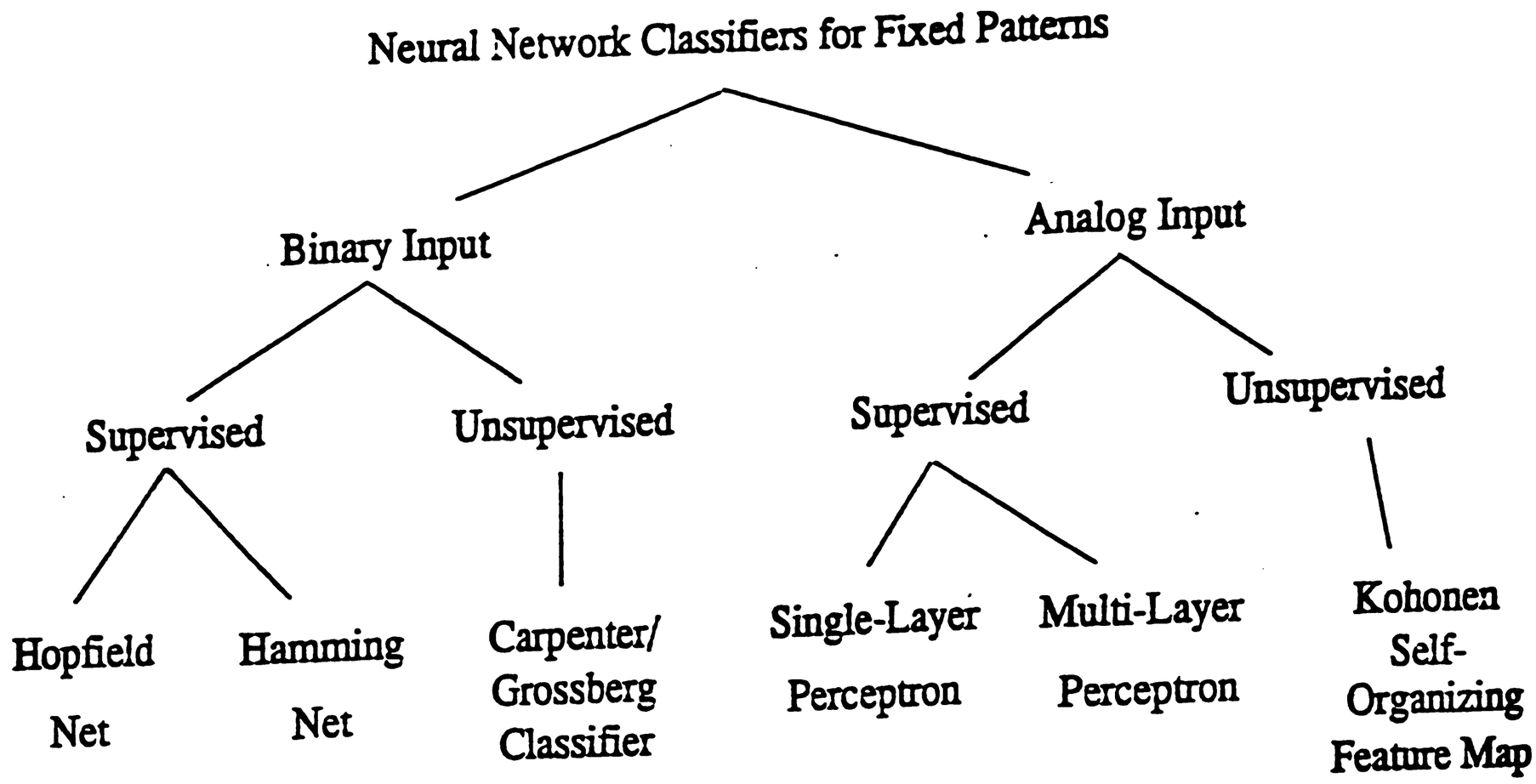


Figure A-1: Taxonomy of the Neural Networks

A.1 Hopfield Net

The Hopfield Net[20] has contributed the surge in the interest in the neural networks. The operation of the Hopfield net of N nodes starts with the assignation of connection weight values as follow:

$$t_{ij} = \begin{cases} \sum_{s=0}^{M-1} x_i^s x_j^s & i \neq j \\ 0 & i=j,\ 0 \leq i,j \leq M-1 \end{cases} \qquad (A.1)$$

where t_{ij} is the connection weight from node i to node j and x_i^s is the ith element of the examplar for class s with the value either 1 or -1. After initialization, the unknown pattern is imposed on the net at time =0 by forcing the output of the net to match the unknown pattern.

$$u_i(0) = x_i, \quad 0 \leq i \leq N-1 \qquad (A.2)$$

where $u_i(t)$ is the output of node i at time t and x_i is the ith element of the input pattern. The net then iterates until the outputs remain unchanged according to the following rule.

$$u_i(t+1) = f_h\left[\sum_{i=0}^{N-1} t_{ij} u_i(t)\right], \quad 0 \leq j \leq M-1 \qquad (A.3)$$

where the function f_h is the hard limiting nonlinearity function. The node outputs then represent the examplar pattern that best matches the unknown input.

The Hopfield net, however, has two major limitations. First, the number of patterns that can be stored and accurately recalled is severely limited by the number of connections and nodes. Second, the Hopfield net becomes unstable

when the examplar pattern shares many bits in common with another examplar pattern.

A.2 The Hamming Net

In a communication system, when binary fixed length signals are transmitted through a memoryless channel, a classifier that calculates the hamming distance to the examplar for each class and select the class that has the minimum hamming distance is called the Hamming Net. The hamming distance is defined as the number of bits in the input which do not match the corresponding examplar bits.

A Hamming Net implementation normally consists of two subnets, the lower subnet layer, and the upper MAXNET layer. The lower subnet is responsible for the calculation of the matching scores from the input and the MAXNET layer selects the node which has the maximum output. If we have N input nodes and M output nodes, the operation of the Hamming Net begins with the weights and the thresholds values in the lower subnet being set in such a way that the output layers of the lower subnet calculates the quantity, the matching score, N minus the Hamming distance to the examplar pattern. The range of the matching score is therefore from 0 to N. The higher the matching score is, the better match it is for the input to the corresponding classes with examplars. The thresholds and weights of the MAXNET subnet are fixed. All thresholds are set to zero and the weights from each node to itself are 1. Weights between nodes are inhibitory with a value of $-\varepsilon$ where $\varepsilon < 1/M$. Therefore, we can express the operation of the Hamming Net as follow

In the lower subnet·

$$W_{ij} = \frac{x^j_i}{2} \quad \theta_j = \frac{N}{2} \tag{A.4}$$

$$0 \le i \le N-1, \quad 0 \le j \le M-1$$

In the upper subnet·

$$t_{kl} = \begin{cases} 1, & k = l \\ -\varepsilon, & k \ne l, \quad \varepsilon < \dfrac{1}{M} \end{cases}$$

$$0 \le k, l \le M-1$$

where W_{ij} is the connection weight from input node i to node j in the lower subnet and θ_j denotes the threshold value of node j, t_{kl} is the connection weight from node k to node l in the upper MAXNET and x^j_i is the ith element of examplar j. After the weight and the threshold values are initialized, a pattern with N elements can then be presented at the bottom of the Hamming Net. The pattern has to be present long enough, though, to allow the calculation of the matching score to be settled.

$$\mu_j(0) = f_t\left(\sum_{i=0}^{N-1} W_{ij}\, x_i - \theta_j\right) \tag{A.5}$$

$$0 \le j \le M-1$$

where μ_j is the output of node j in the lower subnet, x_i is the ith element of the input node and f_t is a nonlinear threshold function. An assumption of the nonlinear function has to be made that the maximum input to the function will not cause the output to saturate. After the net is initialized with the pattern, the pattern can be removed and the MAXNET iterates until the output of only one node is positive. Once this condition is achieved, the classification process is

then complete and the selected class is the one that corresponds to the positive node.

$$\mu_j(t+1) = f_t\left(\mu_j(t) - \varepsilon \sum_{k \neq j} \mu_k(t)\right) \tag{A.6}$$

$$0 \leq j,k \leq M-1$$

In literature, it has been proven that the MAXNET will always converge and find the node with the maximum value when $\varepsilon < \frac{1}{M}$

The Hamming Net has several advantages over the Hopfield Net. First of all, since the Hamming Net implements the optimum minimum error classifier when bit errors are random and independent, it has at least equivalent or even better performance over the Hopfield Net. The Hamming Net also requires less connections than the Hopfield Net. Furthermore, the Hamming Net requires less number of inputs then Hopfield Net. In addition, the Hamming Net does not suffer from spurious output patterns which can produce a non matching result.

A.3 The Carpenter/Grossberg Classifier

Carpenter and Grossberg have designed a net which forms clusters and is trained without supervision.[21] This net implements a clustering algorithm that is similar to the simple sequential leader algorithm. The leader algorithm selects the first input as the examplar for the first cluster. When the second input is present to the net, it compares the second input to its first cluster and computes the corresponding matching score. If the matching score is within a certain threshold value, then the second input is classified as the first cluster, otherwise, it is the examplar for the new cluster. This process is repeated for all

following inputs. Therefore, the number of clusters grows with time and it is dependent upon the matching score and the threshold value set for the net.

Matching scores are computed using feed-forward connections and the maximum value is enhanced using lateral inhibitation among the output nodes. Thus, the structure of Carpenter/Grossberg net is similar to the Hamming net described in the previous section. However, there are distinct differences between both nets. The Carpenter/Grossberg net provides feedback connections from the output nodes to the input nodes. Furthermore, mechanisms are also provided to turn off the output node with a maximum value, and to compare examplars to the input for the threshold test required by the leader algorithm. This net is completely described using nonlinear differential equations including extensive feedback.

The operation of the Carpenter/Grossberg net starts from the initialization of setting all the examplars represented by connection weights to zero. In addition, a matching threshold, called vigilance, ranging from 0 to 1 must be set. The vigilance determines how close a new input pattern must be to a stored examplar to be considered similar. Therefore, a vigilance value near 1 requires a close match between the input pattern and the stored examplar while a smaller value of vigilance accepts less similarity. The initialization can be described as follow

$$
\begin{aligned}
& t_{ij}(0) = 1 \qquad 0 \le i \le N-1 \\
& b_{ij}(0) = \frac{1}{N+1} \qquad 0 \le j \le M-1 \\
& \textit{set } \rho \qquad 0 \le \rho \le 1
\end{aligned}
\tag{A.7}
$$

where $b_{ij}(t)$ is the bottom-up and the $t_{ij}(t)$ is the up-down connection weight

between the input node i and output node j at time t, ρ is the vigilance value. After the input pattern is presented to the bottom of the net, the input pattern is compared to all stored examplars in parallel as in the Hamming net to produce the matching scores as

$$u_j = \sum_{i=0}^{N-1} b_{ij}(t) x_i \quad 0 \le j \le M-1 \tag{A.8}$$

where u_j is the output of the node j and x_i is the ith element of the input pattern. The maximum matching score is selected by lateral inhibitation as

$$u_j^* = max\{u_j\} \tag{A.9}$$

The selected examplar is then compared with the input pattern and the ratio of the dot product of the input pattern and the selected examplar divided by the number of 1 bits in the input is computed. If the ratio is greater than the vigilance threshold, then th input is classified to be the selected examplar. The selected examplar is then updated by performing the logical AND operation between the bits of input pattern and the best matched examplar. If the ratio is less than the vigilance threshold, then the input is considered as a new examplar and is added to the net. Each additional examplar requires one more node and 2N new connections to compute the matching score.

$$\| x \| = \sum_{i=0}^{N-1} x_i \tag{A.10}$$

$$\| T \cdot x \| = \sum_{i=0}^{N-1} t_{ij}^{*} \cdot x_i$$

$$\text{is} \quad \frac{\| T \cdot x \|}{\| x \|} > \rho \, ?$$

YES → *Match the selected examplar*

NO → *New examplar*

and the update of the selected examplar is

$$t_{ij}^{*}(t+1) = t_{ij}^{*}(t) \cdot x_i \tag{A.11}$$

$$b_{ij}^{*}(t+1) = \frac{t_{ij}^{*} x_i}{0.5 + \sum_{i=0}^{N-1} t_{ij}^{*}(t) x_i}$$

There is problem associated with the Carpenter/Grossberg net. This net works well with perfect input patterns; however, even a small noise with the input pattern can cause the net to classify the noisy input pattern as a new examplar. Modifications are necessary to enhance the performance of this algorithm in noise.

A.4 Single Layer Perceptron

The single layer perceptron is the first net that can accept analog as well as the digital inputs. This net has generated much interest initially because of its ability to learn to recognize simple patterns. A perceptron, when first developed, can decides whether the input belongs to one of the two classes, A and B. Therefore, this net can be used as a classifier. The single node computes

a weighted sum of the input elements, subtracts a threshold (θ) and passes the result through a hard limiting nonlinearity such that the output y is either +1 or -1. The output is categorized as class A if the output is 1, while the output is judged as class B if the output is -1. In order to analyze the net behavior, a plot of the decision regions created in the multidimensional space spanned by the input variable can be generated. If there are only two input variables, then the decision region is separated by a line; however, if there are more than two input variables, then the decision region would be separated by a hyperplane.

The connections weights and the threshold value of the net can be either fixed or adaptive. If one chooses to use adaptive weights, then a updating (or learning) algorithm is necessary. The original convergence procedures was developed by Rosenblatt[22]. The net starts its initialization by randomly assigning small non-zero weight values. When an input of N elements is applied to the net, the output is computed as

$$y(t) = f_h\left(\sum_{i=0}^{N-1} W_i(t)x_i(t) - \theta\right) \tag{A.12}$$

where $W_i(t)$ is the ith weight value at time t, f_h is a nonlinear hardlimiting function and θ is the threshold value. If the output of the net $y(t)$ is different from the desired response, $d(t)$, then an error is generated and the weight values needed to be adjusted.

$$W_i(t+1) = W_i(t) + \eta\,[d(t)-y(t)]\,x_i(t) \quad 0 \le i \le N-1 \tag{A.13}$$

where $d(t)$ is +1 if the input is from class A and is -1 if the input is from class B, and η is the gain term or the convergence factor which controls the adaptation rate. This gain term must be adjusted to satisfy the conflicting requirements of

fast adaptation for real changes in the input distributions and averaging of past inputs to provide stable weight estimated. One problem of the perceptron is that the decision boundaries may oscillate continuously when inputs are not separable and distribution overlap. Another problem of this net is the averaging process in the adaptation algorithm which requires not only a complex hardware implementation but is also time and memory space consuming.

A modification to the perceptron convergence procedure can form the least mean square (LMS) error algorithm. This algorithm minimizes the mean square error between the desired response and the actual output of the perceptron. This algorithm is called the Widrow-Hoff algorithm.[13, 11] This thesis employs a modified version of the Widrow-Hoff LMS error learning algorithm. The LMS algorithm is identical to the perceptron convergence procedure described above except the hard limiting function is replaced by a linear function. Weight values are thus corrected on every trial by an amount that depends on the difference between the desired and the actual output, or the error.

The perceptron training algorithm makes no assumptions concerning the shape of underlying distributions but focuses on error that occur where distribution overlap. Therefore, it is more robust than classical techniques and work well when inputs are generated by nonlinear processes and are heavily skewed and non-Gaussian. The adaptation algorithm is simple to implement and it does not require any other information other than the present values of the error and the input variables. However, the perceptron does not work well as a classifier if the classes cannot be separated by a hyperplane.

A.5 Multi-Layer Perceptron

Multi-layer perceptrons are feed-forward nets with one or more layers of nodes between the input and output nodes. These additional layers contain hidden units or nodes that are not directly connected to both the input and output nodes. Multi-layer perceptrons overcome many of the limitations of single-layer perceptron, but were generally not used in the past because effective training algorithm was not available. This problem is recently changed with the newly developed algorithms.[23] The capability of multi-layer perceptron stem from the nonlinearities used within the nodes. If nodes were linear, then a single-layer net with appropriately chosen weights could exactly duplicate those calculations performed by any multi-layer net. While the single-layer perceptron forms half-plane decision regions, a two layer perceptron can form any, possible unbounded, convex region in the space spanned by the inputs. Convex regions are formed from the intersections of the half-plane regions formed by each node in the first layer of the multi-layer perceptron. The number of nodes must be large enough to form a decision region that is as complex as is required by a given problem. On the other hand, it must not be so large that the many weights required cannot be reliably estimated from the available training data.

A three layer perceptron can form arbitrarily complex decision regions. This property depends on the partitioning the desired region into small hypercube (hypersquare if there are only two inputs). Each hypercube requires 2N nodes in the first layer, and one node in the second layer. Hypercubes are assigned to the proper decision regions by connecting the output of each second-layer node only to the output node corresponding to the decision region that node's hypercube is in. The construction procedure can be generalized to use arbitrarily shaped convex regions instead of small hypercubes and is capable of

generating the disconnected and non-convex region. Since three layer net can generate arbitrary decision region, a classifier can be built with no more than three layer net. The number of nodes in the second layer must be greater than one when decision regions are disconnected or meshed and cannot be formed from one convex region. In the worst case, the number of nodes in the second layer is equal to the number of disconnected regions in input distributions. The number of nodes in the first layer should be sufficient enough to provide three or more edges for each convex area generated by every second-layer node. There should thus typically be more than three times as many nodes in the first layer as in the second layer. This analysis applies to the multi-layer perceptron with one output node with hard limiting nonlinear function built-in the output node.

Similar behavior is exhibited by multi-layer perceptron with multiple output nodes when sigmoidal nonlinearities are used and the decision rule is to select the class corresponding to the output node with the largest output. However, the decision region are typically bounded by smooth curves instead of by straight line segments and thus analysis is more difficult. This type of net, however, can be trained with the back-propagation training algorithm. The back-propagation algorithm is a generalization of the LMS algorithm. It uses a gradient decent technique to minimize the function of the mean square difference between the desired and actual outputs. An essential component of the algorithm is the iterative method that propagates error terms required to adapt weights back from nodes in the output layers to nodes in lower layers. In a multi-layer architecture, if the index for the first hidden layer is j, the index for the second hidden layer is k and the output layer index is l, then the output for each layer can be expressed as

$$x'_j = f\left(\sum_{i=0}^{N-1} W_{ij} x_i - \theta_j\right) \quad 0 \le j \le N_1 - 1 \tag{A.14}$$

$$x''_k = f\left(\sum_{j=0}^{N_1-1} W_{jk}' x'_j - \theta'_k\right) \quad 0 \le k \le N_2 - 1$$

$$y_l = f\left(\sum_{k=0}^{N_2-1} W_{kl}'' x_k'' - \theta''_l\right) \quad 0 \le l \le M - 1$$

where $\theta_j, \theta_k, \lambda_l$ are the threshold values for the first hidden layer, second hidden layer and the output layer respectively. The weight values are updated as follow

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta_j x'_i \tag{A.15}$$

where $W_{ij}(t)$ is the weight from hidden node i or from an input node to node j at time t, x'_i is the output of node i or is an input node, and η is the gain factor. Equation (A.15) is a generalized equation of how the weights should be updated. For example, if we focus our analysis between the input and the first hidden layer nodes, then the indices remain unchanged. If we, however, concentrate our analysis between the first hidden layer and the second hidden layer, then we have to change the index i and j in equation (A.15) to j and k. The error term, δ_j, is also a generalized term. If node j is an output node, then

$$\delta_j = y_j(1 - y_j)(d_j - y_j) \tag{A.16}$$

where d_j is the desired output of node j and y_j is the actual output. If we again conform this equation into the index convention in equation (A.14), then the index j in this equation has to be changed to l. If the node j is an internal hidden node then

$$\delta_j = x_j'(1-x_j')\sum_k \delta_k W_{jk} \tag{A.17}$$

where k is over all nodes in the layers above node j. Convergence is sometimes faster if a momentum term is added and weight changes are smoothed by

$$W_{ij}(t+1) = W_{ij}(t) + \eta\,\delta_j x'_i + \alpha(W_{ij}(t) - W_{ij}(t-1)) \quad 0 \le \alpha \le 1 \tag{A.18}$$

The back propagation algorithm, however, may find a local minimum instead of global minimum. In addition, the number of presentations of training data required for convergence has many times been large.

A.6 Kohonen's Self Organizing Feature Maps

Kohonen's algorithm[24] creates a vector quantizer by adjusting weights from common input nodes to M output nodes arranged in a two dimensional grid. Input vectors are presented sequentially in time without specifying the desired output. After enough input training vectors, weights will specify cluster or vector centers that sample the input space such that the point density function of the vector centers tends to approximate the probability density function of the input vectors. Therefore, the weights will be organized such that topologically close nodes are sensitive to inputs that are physically similar. The algorithm that forms feature maps requires a neighborhood, which will slowly decrease in size with time, to be defined around each node. Weights are initially set to small random values just like the single and multi-layer perceptron in previously section. When the input is presented to the map, the distance between the input and all nodes is computed as

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - W_{ij}(t))^2 \qquad (A.19)$$

where $x_i(t)$ is the input to node i at time t, $W_{ij}(t)$ is the weight from input node i to output node j at time t, and d_j is the distance between the input and each output node j. If the weight vectors are normalized to have constant length (i.e. the sum of the squared weight from all inputs to each output are identical), then the node with the minimum Euclidean distance can be found to form the dot product of the input and the weights. This selection can be done with extensive lateral inhibitation as in the MAXNET described in previous section. Once this node is selected, weights to the node and the weights to the neighborhood of the selected node is modified to make these nodes more responsive to the current input. If we define the index of the node selected as j^*, then the weights will be updated as

$$W_{ij}(t+1) = W_{ij}(t) + \eta(t)(x_i(t) - W_{ij}(t)) \qquad (A.20)$$
$$j \in NE_{j^*}(t) \qquad 0 \le i \le N-1$$

where $NE_{j^*}(t)$ is the neighborhood of node j^* and $\eta(t)$ is the gain term ranging from 0 to 1 that decrease in time. Once the weights converge, the weights are then fixed with the gain term is set to 0.

This map can be used as speech recognizer as a vector quantizer. Unlike the Carpenter/Grossberg classifier, this algorithm can perform relatively well because of the limited number of classes available, the slower weight adaptation, and the termination of adaptation once convergence is reached. This algorithm is thus a viable sequential vector quantizer when the number of

clusters desired can be specified before use and the amount of training data is large relative to the number of clusters desired.

Appendix B
Derivation of the Varying Convergence Factor

The convergence factor, μ, in the LMS and the clipped data LMS algorithm plays an important role in the circuit convergence speed. If the convergence factor is too large, then each incremental weight calculated may be too large to make the weights reach their steady state values. As a result, the circuit will overcorrect itself and the error signal will oscillate in amplitude. However, larger convergence factor aids in the convergence speed as the error signal can be reduced in a much faster pace. On the other hand, the small convergence factor will guarantee a better convergence; however, the convergence speed will be much slower. A delicate balance between the convergence speed and the convergence performance requires a careful evaluation when design a adaptive circuit. Therefore, should the convergence factor be variable, large when the error signal is large, reduced accordingly to the error signal, then the circuit performance may be optimized.

This appendix describes the derivation of a variable convergence factor scheme used in the linear adaptive neuron discussed in this thesis. This scheme is the direct result of the SONOS nonvolatile memory transistor synaptic elements. Let us first review the channel conductance of a SONOS transistor, the channel conductance can be written as

$$g_{dsk}^{+}(m) = \beta [V_r - V_{thk}^{+}(m)] \tag{B.1}$$

where k is the spatial index and m is the time index, V_r is the read voltage, V_{th} is the electrically programmable threshold voltage, the superscript + denotes the

channel conductance connected to the positive programming path, β is the beta of the MOS transistor defined as

$$\beta = \mu_{eff}(W/L)\,C_{eff}$$

where the μ_{eff} is the effective mobility of carrier, W and L are the width and the length of the transistor, C_{eff} is the effective gate capacitance. If we increase the time index by one, the channel conductance then becomes

$$g_{dsk}^{+}(m+1) = \beta[V_r - V_{thk}^{+}(m+1)] \tag{B.2}$$

Combining equations (B.1) and (B.2), we can rewrite the channel conductance in the following form

$$\begin{aligned} g_{dsk}^{+}(m+1) &= g_{dsk}^{+}(m) - \beta[V_{thk}^{+}(m+1) - V_{thk}^{+}(m)] \\ &= g_{dsk}^{+}(m) - \beta\,\Delta V_{thk}^{+}(m+1) \end{aligned} \tag{B.3}$$

where $\Delta V_{thk}^{+}(m+1)$ denotes the difference in threshold voltage at adjacent time slot. Equation (B.3) is a general expression for the relationship between channel conductance at different time slot. Therefore, the following relationship can also be drawn

$$\begin{aligned} g_{dsk}^{+}(m) &= g_{dsk}^{+}(m-1) - \beta\,\Delta V_{thk}^{+}(m) \\ g_{dsk}^{-}(m) &= g_{dsk}^{-}(m-1) - \beta\,\Delta V_{thk}^{-}(m) \end{aligned} \tag{B.4}$$

From the analysis in chapter 4, the weight value can be written as the the differential conductance between the positive and negative programming paths multiplies by the gain factor of the circuit as

$$W_k(m+1) = R_f\left(\frac{R_2}{R_1}\right)\left[g_{dsk}^{+}(m) - g_{dsk}^{-}(m)\right] \tag{B.5}$$

Substitute equation (B.1) into (B.5) and assuming the same beta for the transistors, then we can rewrite the weight as

$$W_k(m+1) = R_f\left(\frac{R_2}{R_1}\right)\left\{\beta\left[V_r - V_{thk}^{+}(m)\right] - \beta\left[V_r - V_{thk}^{-}(m)\right]\right\} \tag{B.6}$$

$$= R_f\left(\frac{R_2}{R_1}\right)\left[\beta\left(V_{thk}^{-}(m) - V_{thk}^{+}(m)\right)\right]$$

OR

$$W_k(m) = R_f\left(\frac{R_2}{R_1}\right)\left[\beta\left(V_{thk}^{-}(m-1) - V_{thk}^{+}(m-1)\right)\right]$$

From equation (B.6), the incremental weight can be obtained. Let us concentrate on the clipped data LMS error algorithm implemented in the linear adaptive neuron. The clipped data LMS error algorithm is shown below for easier comparison and comprehension purpose

$$W_k(m+1) = W_k(m) + 2\mu_{clmse}|\varepsilon(m)|sgn(\varepsilon(m))sgn(x_k(m))$$

where μ_{clmse} denotes the convergence factor for the clipped data LMS error algorithm. Combining the above expression and equation (B.6), the following result can be obtained

$$W_k(m+1)-W_k(m) = 2\mu_{clmse}|\varepsilon(m)|sgn(\varepsilon(m))sgn(x_k(m)) \tag{B.7}$$

$$= R_f\left(\frac{R_2}{R_1}\right)\left[\beta\,(V_{thk}^{-}(m)-V_{thk}^{+}(m)) - \beta\,(V_{thk}^{-}(m-1)-V_{thk}^{+}(m-1))\right]$$

$$= R_f\left(\frac{R_2}{R_1}\right)\beta\left[(V_{thk}^{-}(m)-V_{thk}^{-}(m-1)) - (V_{thk}^{+}(m)-V_{thk}^{+}(m-1))\right]$$

OR

$$W_k(m+1)-W_k(m) = R_f\left(\frac{R_2}{R_1}\right)\beta\left[\Delta V_{thk}^{-}(m) - \Delta V_{thk}^{+}(m)\right]$$

Since the result of the digital multiplication, $sgn(\varepsilon(m))\,sgn(x_k(m))$, is either 1 or -1 (1 if both signs are the same polarity and -1 if the signs are opposite in polarity), the following conclusion between the convergence factor and the change of the threshold voltage of the SONOS nonvolatile memory transistors can be drawn

$$\pm\, 2\mu_{clmse}|\varepsilon(m)| = R_f\left(\frac{R_2}{R_1}\right)\beta\left[\Delta V_{thk}^{-}(m) - \Delta V_{thk}^{+}(m)\right] \tag{B.8}$$

From section 4.2 in the thesis, the incremental weight change is achieved by applying the programming voltage to the gate electrode of the SONOS nonvolatile memory transistors. The programming voltage, V_p, can be expressed in one of the either forms below

$$V_p(m) = 2G|\varepsilon(m)| + V_{pdc+} \quad \{Positive\ Programming\ Path\} \tag{B.9}$$

$$V_p(m) = -2G|\varepsilon(m)| - V_{pdc-} \quad \{Negative\ Programming\ Path\}$$

where G is the gain factor in the summing amplifier section and can be expressed as

$$G = R_f \frac{R_2}{R_1}$$

Depending on the result of the digital multiplication, either the positive programming voltage or the negative programming voltage is chosen by the steering network for programming the SONOS devices. I will concentrate the following analysis on the positive incremental weight, while negative incremental weight has almost the same derivation with minor sign changes.

If we differentiate equation (B.8) with respect to the error signal, ε, then the following equation can be obtained

$$2\mu_{clmse} = R_f\left(\frac{R_2}{R_1}\right)\beta\frac{\partial}{\partial\varepsilon}\left[\Delta V_{thk}^{-}(m) - \Delta V_{thk}^{+}(m)\right] \qquad (B.10)$$

Differentiate equation (B.9) with respect to error and substitute the result into equation (B.10), we arrive

$$2\mu_{clmse} = R_f\left(\frac{R_2}{R_1}\right)\beta\, 2G\frac{\partial}{\partial V_p}\left[\Delta V_{thk}^{-}(m) - \Delta V_{thk}^{+}(m)\right]$$

thus

$$\mu_{clmse} = R_f\left(\frac{R_2}{R_1}\right)\beta\, G\frac{\partial}{\partial V_p}\left[\Delta V_{thk}^{-}(m) - \Delta V_{thk}^{+}(m)\right] \qquad (B.11)$$

If we make an approximation as follow

$$\Delta V_{thk}^{-}(m) = -\Delta V_{thk}^{+}(m) = \Delta V_{thk}(m)$$

then we can rewrite the convergence factor as

$$\mu_{clmse} = R_f\left(\frac{R_2}{R_1}\right)\beta\, G\,\frac{\partial}{\partial V_p}\left[\Delta V_{thk}(m)\right] \tag{B.12}$$

The following analysis is based on the MNOS device structure which exhibits nonvolatile memory behavior as the SONOS device structure. The change in threshold voltage, ΔV_{thk}, can be written in the following form[25]

$$\Delta V_{th} = V_T\left(1+\frac{C_o}{C_N}\right)ln\left(1+\frac{t_p}{\tau}\right) \tag{B.13}$$

where t_p is the programming pulse width, C_o and C_N are the oxide and nitride capacitance defined as

$$C_o = \frac{\varepsilon_o}{x_o} \quad C_N = \frac{\varepsilon_N}{x_N}$$

where ε_o, ε_N are the permittivity of the oxide and nitride, x_o, x_N are the oxide and nitride thicknesses respectively, V_T is the characteristic tunnel oxide voltage and τ is the characteristic time written as

$$\tau = V_T\left(\frac{C_o+C_N}{J_T}\right)exp\left(\frac{-V_o}{V_T}\right) \tag{B.14}$$

where J_T is the tunneling current density and V_o is the voltage drop across the tunneling oxide

$$V_o = \left(\frac{x_o}{x_{eff}}\right) V_p$$

$$x_{eff} = x_o + \left(\frac{\varepsilon_o}{\varepsilon_N}\right) x_N = \left(\frac{x_o}{\varepsilon_o} + \frac{x_N}{\varepsilon_N}\right) \varepsilon_o$$

$$= \left(\frac{1}{C_o} + \frac{1}{C_N}\right) \varepsilon_o = \left(1 + \frac{C_o}{C_N}\right) x_o$$

From the relationship shown above, we can draw the following results

$$\frac{\partial \Delta V_{th}}{\partial V_p} = \frac{\partial \Delta V_{th}}{\partial V_o} \left(\frac{x_o}{x_{eff}}\right) \tag{B.15}$$

Substitute equation (B.13) into (B.15), then

$$\frac{\partial \Delta V_{th}}{\partial V_p} = \left(\frac{x_o}{x_{eff}}\right) \frac{\partial}{\partial V_o} \left[V_T \left(1 + \frac{C_o}{C_N}\right) ln \left(1 + \frac{t_p}{V_T \left(\frac{C_o + C_N}{J_T}\right) exp \left(\frac{-V_o}{V_T}\right)} \right) \right]$$

$$= \frac{x_o}{x_{eff}} \left[\frac{V_T \left(1 + \frac{C_o}{C_N}\right) \frac{t_p J_T}{V_T (C_o + C_N)} \frac{1}{V_T} exp \left(\frac{-V_o}{V_T}\right)}{1 + \frac{t_p}{\tau}} \right] \tag{B.16}$$

$$= \frac{x_o}{x_{eff}} \left[\frac{\left(1 + \frac{C_o}{C_N}\right) \frac{t_p}{\tau}}{1 + \frac{t_p}{\tau}} \right]$$

$$= \frac{\frac{t_p}{\tau}}{1 + \frac{t_p}{\tau}}$$

Therefore, we can rewrite the convergence factor as follow

$$\mu_{clmse} = 2R_f \frac{R_2}{R_1} \beta G \frac{\frac{t_p}{\tau}}{1 + \frac{t_p}{\tau}} \tag{B.17}$$

If we rearrange equation (B.17) and substitute the expression for τ, then the following result may be obtained

$$\mu_{clmse} = \frac{\mu_{clmse}(0) \frac{t_p J_T}{V_T(C_o+C_N)} \exp\left(\frac{V_p C_N}{V_T(C_o+C_N)}\right)}{1 + \frac{t_p J_T}{V_T(C_o+C_N)} \exp\left(\frac{V_p C_N}{V_T(C_o+C_N)}\right)} \tag{B.18}$$

where $\mu_{clmse}(0)$ is a constant contained the gain factor of the circuit as

$$\mu_{clmse}(0) = 2\beta G R_f \frac{R_2}{R_1}$$

To further simplify the convergence factor expression, we can rewrite equation (B.18) in the following form

$$\mu_{clmse} = \frac{\mu_{clmse}(0) \mathrm{K}\, e^{\alpha|\varepsilon|}}{1 + \mathrm{K}\, e^{\alpha|\varepsilon|}} \tag{B.19}$$

where

$$\mathrm{K} = \frac{t_p J_T}{V_T(C_o+C_N)} \exp\left[\frac{V_{pdc}}{\left(1+\frac{C_o}{C_N}\right)V_T}\right]$$

$$\alpha = \frac{2G}{\left(1+\frac{C_o}{C_N}\right)V_T} \tag{B.20}$$

Notice from equation (B.19) that when the error signal is large at the

beginning of the adaptation process, the convergence factor has the value of $\mu_{clmse}(0)$; while the error signal starts to reduce due to the error minimization algorithm, the convergence factor also starts to reduce in value. Therefore, the speed of convergence depends not only on the gain factor of the circuit, but also the SONOS device parameters. Equations (B.19) and (B.20) also provide the first cut information for what device parameters will provide the best convergence performance for optimum circuit operation.

Appendix C
Software Simulation of the Linear Adaptive Neuron

This appendix presents a software simulation program designed to simulate the behavior of the linear adaptive neuron under different learning algorithms. The program is written in the form of a subroutine and is embedded in a larger software package called FIDDLER, written by Dr. Richard Booth. The program utilizes the circuit component values and device parameters as inputs and thus it can simulate the circuit operation as close as possible. The variable convergence factor scheme discussed in the previous appendix is also incorporated in the program. One of the benefits in using the software simulation of the linear adaptive neuron circuit is the availability of the weight changing information with respect to time. The program is written in HP Basic language and the source code is listed below

```
11449   SUB Fiddleneur
11455   ##################################################
11458   Fiddleneur_:OPTION BASE 1
11461   Routine36:!_____________________________________________________
11464   COM/Datastuff/Title$(*),Cname$(*),Generator(*),Datarray(*)
11467   COM/Datastuff/Nchan_max,Npts_max,Nchan,Npts
11470   COM/Character/Blank$,Uline$,Bold$,Norm$,Cr$,Ff$
11473   COM/Character/Wht$,Red$,Yel$,Grn$,Cya$,Blu$,Mag$,Blk$
11476   COM/Leveldata/Kp,Kc,Version$
11479   COM/Constants/C0,E0,Qe,Me,Kb,Eox,Esi,Mec,Mhc,Vt,Ni,Vsat,Egap
11482   DIM Vbfile$[30],File$[30],Drive$[30],L$(30)[200]
11483   OFF KEY
11485   Kp=Kp+1          ! INCREMENT KEY PRIORITY
11486   Okc=Kc           ! SAVE LAST KEY COLOR
11487   Kc=8             ! KEY COLOR
11488   Colormode        ! COLORGRAPH MODE
11489
        _____________________________________________________
11490   ! USER SPECIFIC CODE - PUT DATA INTO:
11491   ! 1> NCHAN                 = NUMBER OF CHANNELS <= NCHAN_MAX
11494   ! 2> NPTS                  = NUMBER OF POINTS IN EACH CHANNEL
11497   ! 3> TITLE$(1)             = TITLE
11500   !          (2)             = SUBTITLE
```

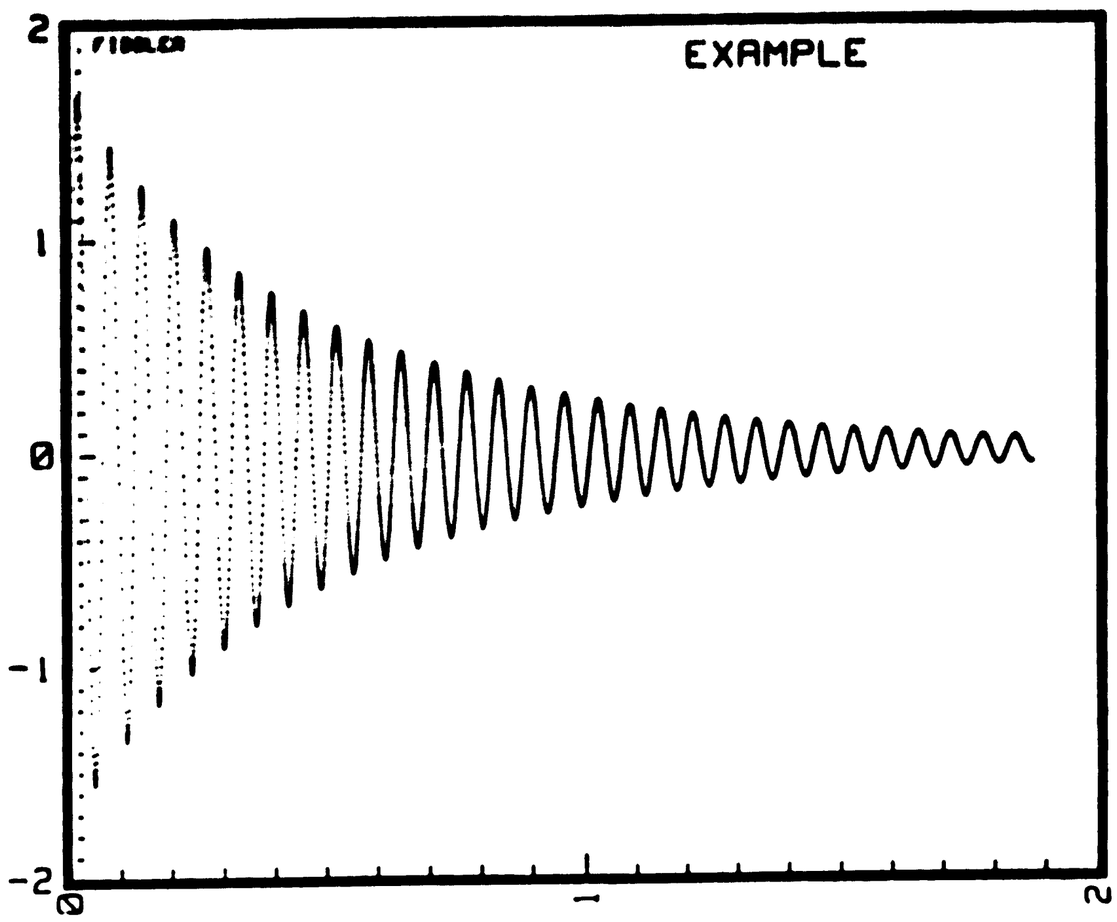
```
11503   !          (3)            = COMMENTS
11506   ! 4> CNAME$(CHAN)         = CHANNEL NAME
11509   ! 5> DATARRAY(PT#,CHAN)   = DATA
11512   !vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
11525   Fund_const(300)
11535   Nchan=12
11545   Npts=Npts_max
11546   ALLOCATE Signal(Npts),X1(Npts),X2(Npts),Desire(Npts),Y(Npts)
11548   !________________________________________________________
11549   Formr1: IMAGE 8A,MD.5DE
11550        DATA "n RF          = 1E4               "
11551        DATA "n R2/R1       = 10                "
11552        DATA "n SIG. AMPL = .2                  "
11553        DATA "n DES. AMPL = 2                   "
11554        DATA "n SIG. FREQ = 100                 "
11555        DATA "n DES. FREQ = 100                 "
11556        DATA "n PHASE DIF = 45                  "
11557        DATA "n GAIN        = .6                "
11558        DATA "n W/L         = 1.5               "
11559        DATA "n MOBILITY   = 400                "
11560        DATA "n TOX         = 2E-7              "
11561        DATA "n XN          = 4.5E-7            "
11562        DATA "n VT (MNOS) = .8                  "
11563        DATA "n JT          = 1E-6              "
11564        DATA "n TP          = 1E-3              "
11565        DATA "n V_PDC       = 5                 "
11566   !________________________________________________________
11567   RESTORE
11568   Nlab=16
11569   FOR I=1 TO Nlab
11570   READ L$(I)
11571   NEXT I
11572   GOSUB Getpars
11573   Axes_flag=1
11586   !________________________________________________________
11587   OFF KEY
11588   Kp=Kp+1
11589   Okc=Kc
11590   Kc=9
11591   Colormode
11593   ON KEY 0 LABEL "SIMU.  PAR'S    ",Kp GOSUB Spar
11594   ON KEY 1 LABEL "SIMU.  START    ",Kp GOSUB Simu
11595   ON KEY 2 LABEL "                ",Kp GOTO Wait
11597   ON KEY 3 LABEL "                ",Kp GOTO Wait
11598   ON KEY 4 LABEL "                ",Kp GOTO Wait
11599   ON KEY 5 LABEL "                ",Kp GOTO Wait
11600   ON KEY 6 LABEL "                ",Kp GOTO Wait
11601   ON KEY 7 LABEL "                ",Kp GOTO Wait
11602   ON KEY 8 LABEL "                ",Kp GOTO Wait
11603   ON KEY 9 LABEL "QUIT   <SIMU.>  ",Kp GOTO Quit
11604   Wait:   GOTO Wait
11605   Quit:   BEEP 5000,.01
11606   DEALLOCATE Signal(*),X1(*),X2(*),Desire(*),Y(*)
```

```
11608   Clearscreen
11609   Kp=Kp-1
11610   Kc=Okc
11611   Colormode
11612   SUBEXIT
11613   !_____________________________________________________________
11614   Spar:    BEEP 5000,.01
11615   GCLEAR
11616   CALL Entry(L$(*),Nlab,"SIMULATION PARAMETERS:")
11617   GOSUB Getpars
11618   RETURN
11619   Getpars:
11620   R_f=VAL(L$(1)[15,29])
11621   R2_over_r1=VAL(L$(2)[15,29])
11622   Sampl=VAL(L$(3)[15,29])
11623   Dampl=VAL(L$(4)[15,29])
11624   Sfreq=VAL(L$(5)[15,29])
11625   Dfreq=VAL(L$(6)[15,29])
11626   Phase=VAL(L$(7)[15,29])
11627   Gain=VAL(L$(8)[15,29])
11628   W_over_l=VAL(L$(9)[15,29])
11629   U_n=VAL(L$(10)[15,29])
11630   T_ox=VAL(L$(11)[15,29])
11631   X_n=VAL(L$(12)[15,29])
11632   V_t=VAL(L$(13)[15,29])
11633   J_t=VAL(L$(14)[15,29])
11634   T_p=VAL(L$(15)[15,29])
11635   V_pdc=VAL(L$(16)[15,29])
11636   RETURN
11637   Simu:    BEEP 5000,.01
11638   DISP Cya$;Bold$;"SIMULATION IN PROGRESS!";Norm$;Grn$
11639   WAIT 2
11640   DISP
11641   Title$(1)="EXAMPLE"
11642   Title$(2)="EXAMPLE"
11643   Title$(3)="EXAMPLE"
11644   Cname$(1)="TIME"
11648   Cname$(2)="W1"
11649   Cname$(4)="W2"
11650   Cname$(5)="Error"
11653   Cname$(6)="EFFECTIVE MU"
11654   ! ##############################################################
11655   K_ox=3.9
11656   Epsilon_0=8.854E-14
11657   K_n=6.5
11658   Cox=K_ox*Epsilon_0/T_ox
11659   Cn=K_n*Epsilon_0/X_n
11660   Ceff=Cox*Cn/(Cox+Cn)
11661   Beta=U_n*W_over_l*Ceff
11662   U_eff_0=2*Gain*Beta*R_f*R2_over_r1
11663   K=T_p*J_t/(V_t*(Cox+Cn))*EXP(V_pdc/((1+Cox/Cn)*V_t))
11664   Alph=2*Gain/((1+Cox/Cn)*V_t)
11665   Phase=Phase*PI/180                    ! RADIANS
```

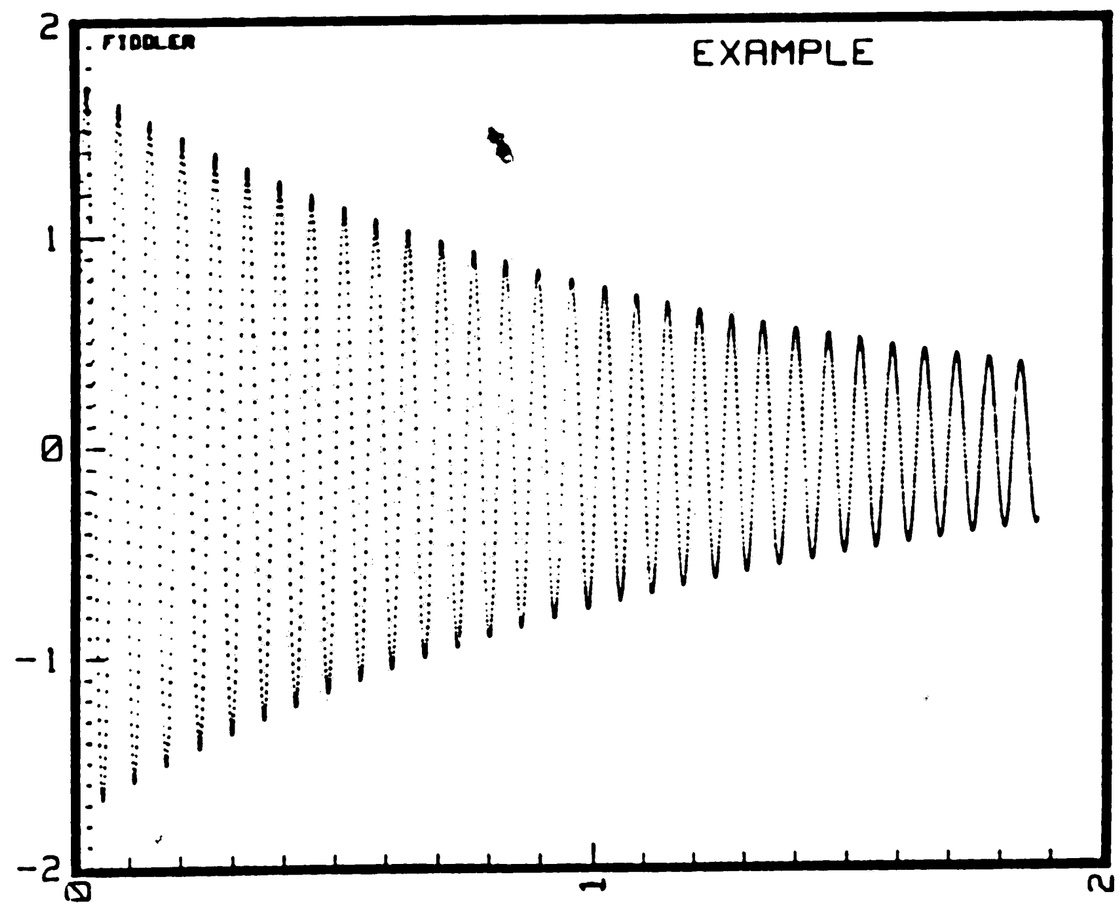
```
11666   Dt=(1/Sfreq)*.05                          ! PERIOD/20(TIME STEP)
11667   T=0                                       ! TIME
11668   !*********************** WEIGHTS INITIALIZATIONS *******
11670    W1=20                                     ! FIRST WEIGHT
11671    W2=20                                     ! SECOND WEIGHT
11672    FOR I=1 TO Npts
11673    Signal(I)=Sampl*SIN(Sfreq*T)
11675    Desire(I)=Dampl*SIN(Dfreq*T+Phase)
11676    X1(I)=Signal(I)                    ! SAMPLE INPUT SIGNAL
11677    X2(I)=Sampl*SIN(Sfreq*T-PI/2)
11678    Y(I)=X1(I)*W1+X2(I)*W2                        ! CONVOLUTION
11680    Err=Desire(I)-Y(I)
11681    Mu_eff=U_eff_0*K*
               EXP(Alph*ABS(Err))/(1+K*EXP(Alph*ABS(Err)))
11682    PRINT "I, MU_EFF",I,Mu_eff
11685    W1=W1+2*Mu_eff*Err*X1(I)          ! UPDATE WEIGHTS
11686    W2=W2+2*Mu_eff*Err*X2(I)
11687    Datarray(I,1)=T
11688    Datarray(I,2)=W1
11689    Datarray(I,4)=W2
11690    Datarray(I,5)=Err
11691    Datarray(I,6)=Mu_eff
11692    T=T+Dt
11693    NEXT I
11696   !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
11697    Outuser:!
11698    Genval
11699    BEEP 5000,.01
11700    DISP Cya$;Bold$;"FINISHED !!!";Norm$;Grn$
11701    WAIT 2
11702    DISP
11703    Kc=Okc
11704    Colormode
11705    Kp=Kp-1
11706    RETURN
11707    SUBEND
```

The effects of the variable convergence factor are found to be significant for the convergence behavior of the circuit. Figure C-1a and b show the circuit convergence performance (error versus time characteristics) under two different schemes: one with a variable convergence factor and the other with a fixed convergence factor.

From figures C-1 a and b, the scheme with the variable convergence factor converges in a much faster rate as postulated in the previous appendix. From the simulation result, the convergence factor will reach its steady state value in



(a)



(b)

Figure C-1: Simulated Convergence Behavior of the Linear Adaptive Neuron with W_0=10 and W_1=-10 (a) Variable Convergence Factor (b) Fixed Convergence Factor

the range of 0.03-0.06. Therefore, more experiments have to be performed with higher convergence factor by increasing the gain of the error feedback path to determine whether the system time constant (or the convergence behavior) will benefit from the results of the software simulation.

Vita

Chun-Yu Malcolm Chen was born August 11, 1967 in Taipei, Taiwan to Cheng-Hsiung and Chin-Chu Wu Chen. He attended Lehigh University from August 1985 to August 1988, graduated with highest honor, and earned a bachelor degree in Electrical Engineering. He has been enrolled as a full time graduate student as Lehigh University since his graduation in the department of Computer Science and Electrical Engineering. He is a student member of IEEE and was elected to the honor societies of Eta Kappa Nu, Tau Beta Pi, and Phi Eta Sigma.